НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій Стіренко

“_____” _____ 2020р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп’ютерні системи та мережі»
спеціальності 123 «Комп’ютерна інженерія»
на тему: «Система моніторингу користувачів бездротових мереж Wi-Fi»

Виконав: студент 4 курсу, групи ІО-63
Булах Антон Олегович _____
(підпис)

Керівник:
ст. викладач Алещенко Олексій Вадимович _____
(підпис)

Консультант з нормо-контролю:
проф. д.т.н. Сімоненко Валерій Павлович _____
(підпис)

Рецензент:
Радченко Костянтин Олександрович _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

(підпис)

Київ - 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 123 «Комп'ютерна інженерія»
Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

"____" _____ 2020р.

ЗАВДАННЯ
на дипломний проєкт студенту
Булаха Антона Олеговича

1. Тема проєкту «Система моніторингу користувачів бездротових мереж Wi-Fi», керівник проєкту Алещенко Олексій Вадимович, ст. викладач., затверджені наказом по університету від «7» травня 2020 р. № 1081-с
2. Термін здачі студентом закінченого проєкту 15 червня 2020р
3. Вихідні дані до проєкту: технічна документація. теоретичні дані, прототип робочого пристрою та додатку
4. Зміст пояснювальної записки: опис предметної області, дослідження аналогів, дослідження засобів написання системи, інструкція користувача
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): граф залежності клієнтського додатку, структурна схема системи, алгоритм бінарного пошуку

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормо-контроль	Сімоненко В. П. проф.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	01.09.2019	
2.	Вивчення та аналіз завдання	05.09.2019	
3.	Розробка архітектури та загальної структури системи	07.10.2019	
4.	Розробка структур окремих підсистем	30.10.2019	
5.	Програмна реалізація системи	19.12.2019	
6.	Оформлення пояснювальної записки	01.05.2020	
7.	Передзахист	29.05.2020	
8.	Захист	15.06.2020	

Студент Антон БУЛАХ

Керівник Олексій АЛЕЩЕНКО

Анотація

В дипломній роботі бакалавра запропонована та реалізована система моніторингу пристроїв, що мають вбудований модуль Wi-Fi. Розроблена система дозволяє збирати статистичну інформацію щодо пересування мас людей що користуються такими пристроями, що в сучасному світі наближено до інформації щодо пересування мас людей у цілому. Детально спроектована архітектура системи та процедура отримання, зберігання та обробки цих даних. Схема розробленої системи ілюструється. Наведені приклади роботи реалізованої системи.

Система має дуже простий алгоритм встановлення та має відповідний графічний інтерфейс. Програмну частину було створено на мовах Python та JavaScript у візуальному середовищі IntelliJ IDEA.

Annotation

In this work for a Bachelor's Degree, a system for monitoring devices with a built-in Wi-Fi module is proposed and implemented. The developed system allows to collect statistical information about the movement of masses of people who are using such devices, which in the modern world is close to the information about the movement of people in general. The system architecture and procedure for obtaining, storing and processing this data are designed in detail. The scheme of the developed system is illustrated. Examples of the implemented system are given.

The system has a very simple installation algorithm and has a corresponding graphical interface. The software was created in Python and JavaScript in the IntelliJ IDEA visual environment.

ОПИС АЛЬБОМУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.467200.001 ВП	Відомість проєкту	1	
3	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.467200.003 ПЗ	Система моніторингу користувачів бездротових мереж Wi-Fi	56	
5	А4	ІАЛЦ.467200.004 Д1	Граф залежності клієнтського додатку	1	
6	А4	ІАЛЦ.467200.005 Д2	Структурна схема системи	1	
7	А4	ІАЛЦ.467200.006 Д3	Алгоритм бінарного пошуку	1	
8	А4	ІАЛЦ.467200.007 Д4	Лістинг програми	6	
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					

Змн.	Арк.	ПІБ	Підпис	Дата	ІАЛЦ.467200.001 ВП		
Розробив		Булах А.О.			Система моніторингу користувачів бездротових мереж Wi-Fi Відомість проекту	Літ.	Арк. 1 / Акрушів 1
Перевірив		Алещенко О.В.					
Н/Контр.		Сімоненко В.П.				КПІ ім. Ігоря Сікорського ФІОТ ІО-63	
Зав.каф.		Стіренко С.Г.					

ТЕХНІЧНЕ ЗАВДАННЯ

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється для використання у курсі «Інженерія програмного забезпечення».

Область застосування: приклад при розробці схожих систем під час вивчення курсу «Інженерія програмного забезпечення».

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського»

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи моніторингу користувачів бездротових мереж Wi-Fi

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література по комп`ютерним системам, публікації в періодичних виданнях, довідники на публікації в Інтернеті щодо даної теми.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Самостійність курсу – чіткі контури предмету вивчення.
- Самодостатність – курс повинен містити необхідну інформацію та дані, які дозволяють в повній мірі розкрити мету курсу
- Коректність та актуальність інформації, яку охоплює даний курс.

5.2. Вимоги до програмного забезпечення

- Unix-подібна ОС або Windows 7/8/10
- Python 3.6+
- SQLite

5.3. Вимоги до апаратної частини

- Серверний комп'ютер на базі процесора Intel Core i3 та вище
- Не менше 1 Гбайт оперативної пам'яті
- Не менше 1 Гбайт простору жорсткого диску

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	01.09.2019
Вивчення та аналіз завдання	05.09.2019
Розробка архітектури та загальної структури системи	07.10.2019
Розробка структур окремих підсистем	30.10.2019
Програмна реалізація системи	19.12.2020
Оформлення пояснювальної записки	01.05.2020
Передзахист	29.05.2020
Захист	15.06.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Система моніторингу користувачів бездротових мереж Wi-Fi»

Київ – 2020 р

ЗМІСТ

Список скорочень

MAC – Media Access Control, унікальний ідентифікатор, який присвоюється кожній одиниці мережевого обладнання

IP – Internet Protocol, протокол мережевого рівня для передавання датаграм між мережами

UDP – User Datagram Protocol, один із протоколів в стеку TCP/IP, що працює без встановлення з'єднання

LAN – Local Area Network, локальна комп'ютерна мережа

Wi-Fi – Wireless Fidelity, загальновживана назва для стандарту IEEE 802.11 передачі цифрових потоків даних по радіоканалах

HTTP – Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів

TLS – Transport Layer Security, криптографічний протокол, що надає можливості безпечної передачі даних в Інтернет

EventStream – технологія відправки повідомлень від сервера до веб-браузеру

HTML – HyperText Markup Language, мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет

JSON – JavaScript Object Notation, текстовий формат обміну даними між комп'ютерами

JavaScript – динамічна, об'єктно-орієнтована, прототипна мова програмування

Rollup – збірник модулей та бібліотек для JavaScript

Svelte – клієнтський фреймворк для створення графічних інтерфейсів на JavaScript

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією

Pip – система керування бібліотеками Python

SQLite – полегшена реляційна система керування базами даних

Scapy – бібліотека Python для взаємодії з бездротовими мережевими інтерфейсами

ВСТУП

За останні роки, чи мабуть, десятки років, розвиток технологій, та, зокрема, мобільних технологій, набрав високого темпу та не збирається спадати. Майже кожна людина зараз має у кишені прилад, який надає їй доступ до знань усього людства, можливість спілкування будь з ким на більшій частині планети, знімати високоякісні фото та відео, читати книжки, статті тощо.

Цим стрімким розвитком вже користуються технологічні гіганти, на кшталт Google чи Facebook, отримуючи від користувачів своїх продуктів – тобто від більшості користувачів мережі Інтернет узагалі – безліч корисної та цінної статистичної й персональної інформації.

Якщо для отримання будь-якої персональної інформації потрібно хоча б отримувати згоду користувача, від збору знеособленної статистичної інформації неможливо навіть відгородитися, не кажучи вже про дозволи і таке інше. Тим паче така інформація не може нести значної шкоди і більшість людей не мають нічого проти такого аналізу їх поведінки чи пересування.

Метою цієї роботи є реалізація одної з можливостей збору такої інформації, а саме – пасивного сканування радіопростору для заміру часу з'явлення унікальних пристроїв навколо скануючого пристрою. При наявності декількох скануючих пристроїв створюється можливість побудови моделей пересування мас людей, що може мати значну цінність для таких установ або місць, що з такими масами стикаються.

Існує декілька пропрієтарних платформ, що здатні на таке сканування, але вони вимагають ліцензій, коштів, контрактів, у таких систем складний та витратний алгоритм розгортання та підтримки працездатності системи та власне збору інформації.

Тому ця робота перш за все спрямована на:

- простоту встановлення та конфігурації;

- відкритий та доступний для будь-кого код та платформу;
- легку маштабованість;
- здатність розгорнути таку систему будь-де та за будь-яких умов.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Постановка задачі

Сучасні інформаційно-комунікаційні технології надають все більше можливостей для усебічної автоматизації, небачених раніше можливостей зв'язку, як завгодно складної обробки інформації та ін. І в той же час зі зростанням потужностей цифрових апаратних засобів, що дозволяють робити все те, що описано у попередньому розділі, зростають все більше і більше загрози до приватного життя широкого кола людей. Якщо раніше коштовні засоби шпіонської відео зйомки загрожували лише знаменитостям, то вже сьогодні кожна особа може стикнутися із прихованим записом своєї життєдіяльності на дешеві мікрокамери (які часто ще і оснащуються виносними об'єктивами типу pinhole, що взагалі дуже важко розпізнати нетренованим оком). Те ж саме стосується і контролю місцеположення суб'єктів у наш час цифрової ери мобільних технологій: із розвитком комунікаційних технологій з'явилося дуже багато способів відстеження місцезнаходження особи, причому як незаконних (але порівняно доступних пересічним зловмисникам), так і цілком законних, що мають право використовувати працівники силових структур та відомств.

Відмітимо, що фіксація місцеположення особи не завжди є однозначно негативним явищем. Навпаки, часто цю можливість можна використати і на благо людини, наприклад, забезпечуючи роботу систем доповненої реальності, які можуть радити, куди поруч можна піти відпочити, де і за скільки купити, що тут поблизу відвідати.

Традиційними методами визначення місцеположення особи є оцінювання місцезнаходження її мобільного телефону. Раніше (до повсюдного впровадження мобільної персональної техніки) відстеження положення суб'єкта було можливим тільки за умови встановлення на ньому (на одязі чи

взутті, сумці чи інших особистих речах) жучків, але зараз такої потреби просто не існує, адже усі люди носять із собою мобільні телефони, які до того ж, в останні роки, майже всі оснащуються такими технологіями, що дозволяють встановити положення власника дуже простими силами (навіть і без залучення спецслужб та урядових організацій).

В першу чергу, для цього використовується мережа супутників GPS (Global Positioning System, тобто глобальної системи позиціонування) - найчастіше, або її аналоги (ГЛОНАСС – російського виробництва, Galileo – Євросоюзу, Beidou – китайського, QZSS – японського та IRNSS – індійського). Однак, деякі мобільні пристрої не оснащуються модулями GPS, що в першу чергу відноситься до планшетів та ноутбуків (а також окремих моделей смартфонів), і це є недоліком самої ідеї використання GPS для позиціонування людей, що мають персональну мобільну техніку.

По-друге, можливим є використання відомих з давніх-давен галузі радіозв'язку методів тріангуляції та трилатерації, що є можливими при постійному знаходженні мобільного пристрою у контакті із системою фіксованих базових станцій стільникового зв'язку. Метод підходить для усіх пристроїв оснащених модулями GSM, що також як і попередній, відсікає ноутбуки та більшість планшетів. Крім того, робота цих методів у стінах приміщень навряд чи буде досить точною (через неможливість урахування загасання сигналів при проходженні через стіни, коридори, вікна-двері і т.п.), і буде надавати похибки порядку десятків, а то і сотень (у великих, розгалужених приміщеннях) метрів.

Таким чином, існує потреба в організації інших варіантів визначення місцеположення особи по його мобільному пристрою і це дійсно можна зробити з використанням технології Wi-Fi (Wireless Fidelity, або «бездротова точність»). Мова іде про позиціонування у великій мережі, яка має значну кількість бездротових роутерів (хоча би десяток і більше). Такі мережі зазвичай розгортаються у торгівельних центрах, великих навчальних закладах,

багатоповерхових офісних будівлях, аеропортах, і т.п. Можливість визначення, де саме у великій будівлі зараз знаходиться підлеглий працівник є дуже корисною, так само, як і працівникові зручно орієнтуватися у віддаленій частині багатоповерхової будівлі, коли система видає йому його поточне місцеположення з підказками, куди саме слід повертати та рухатися.

Відповідно до інформації деяких джерел (наприклад, [1]), виробники Wi-Fi-інфраструктури та провайдери Wi-Fi-послуг давно навчилися позиціонувати Wi-Fi-пристрої в реальному часі з точністю аж до 1 м.

Відмітимо, що подібні системи, які реалізовані у більшій, чи меншій мірі, існують, але, звичайно, мають деякі особливості, які за певних обставин можна вважати недоліками, тому слід розглянути їх докладніше.

Системи Wi-Fi-позиціонування можуть використовуватися власниками торгівельних центрів, аеропортів, стадіонів і метрополітенів для збору і аналізу даних про користувачів системи. Ця інформація може застосовуватися міськими Wi-Fi-мережами, які в сучасних мегаполісах, наприклад, охоплюють майже весь суспільний простір міста.

1.2 Аналіз існуючих аналогів

Одною з найбільших та одночасно високотехнологічних Wi-Fi мереж у Європі є інфраструктура, зведена для пасажирів Московського метрополітену компанією MaximaTelecom – рис.1.1.

Рис. 1.1 Візуалізація системи, що позиціонує користувачів Московського метрополітену

Наведена система є одним із прикладів конкретної реалізації ідей Wi-Fi позиціонування. З іншого боку, існує навіть окрема абревіатура (щоправда, яка має і інші варіанти розшифровування, причому такі, що належать до галузі бездротових мереж, тому при її використанні слід бути обережним) – WPS (Wi-fi Positioning System), що уособлює усі такі системи, тобто символізує саму технологію Wi-Fi позиціонування.

Великим провайдером WPS є всесвітньо відома компанія Skyhook Wireless (Бостон, США) – рис. 1.2, [2].

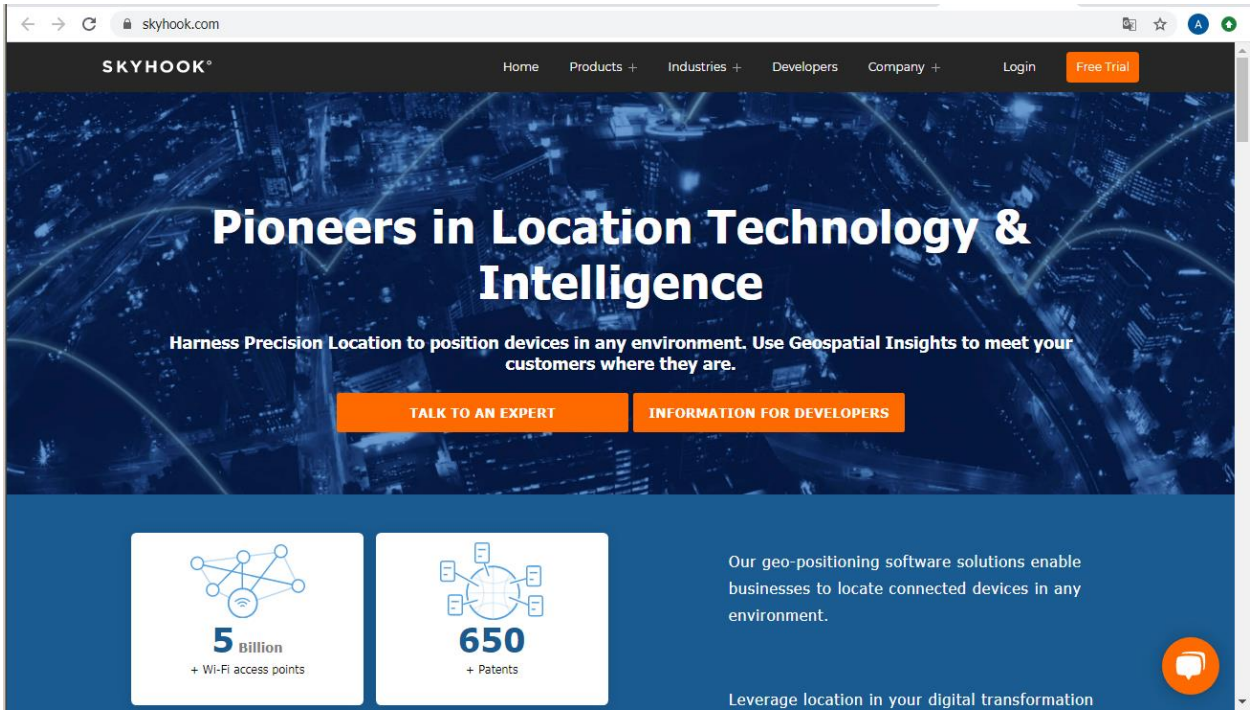


Рис. 1.2 Титульна сторінка сайту компанії Skyhook, що є світовим лідером у галузі WPS

Уважне дослідження сайту компанії дозволяє зробити висновок, що від технології виключно WPS вона поступово перейшла до більш широкого профілю – гібридного позиціонування. Як написано на сайті компанії, при цьому використовуються відомості від трьох джерел геоданих:

- від мереж Wi-Fi;
- від супутників GPS;
- від станцій стільникового зв'язку GSM.

Ідеї гібридного позиціонування є досить популярними на сьогоднішній день у IT-спільноті, ймовірно, через дуже широку доступність відповідних пристроїв, що дозволяють проводити таке позиціонування (Wi-Fi-модулі, GPS-приймачі та модулі GSM). Ще одним із лідерів цього напряму є компанія AlterGeo – рис. 1.3, яка використовує для позиціонування технології Wi-Fi, WiMAX, GSM, LTE.

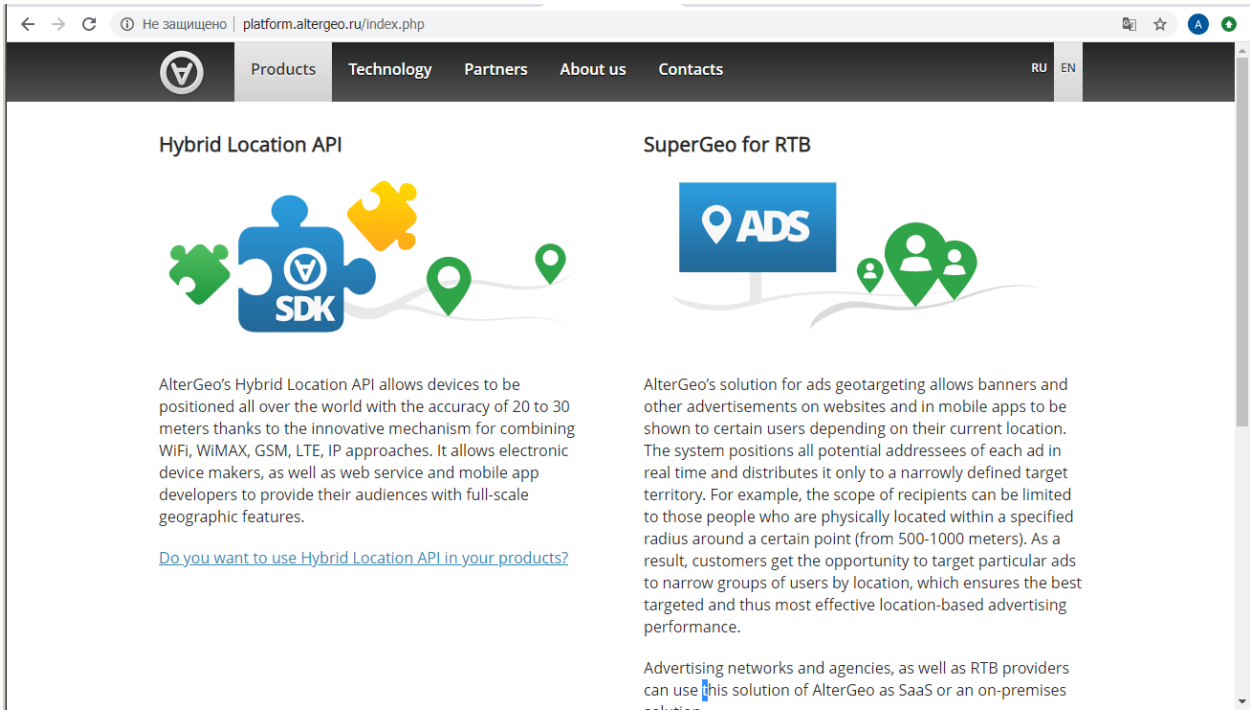


Рис. 1.3 Титульна сторінка системи AlterGeo гібридного позиціонування (що включає WPS)

Відмітимо, що якраз у великих приміщеннях (ТРЦ, ЗВО, тому ж метрополітені, і т.д.) технології саме Wi-Fi-позиціонування працюють на безальтернативній основі, в той час, як, наприклад, на стадіоні чи у міському середовищі щільного Wi-Fi-покриття, більш доцільним видається комбіноване, гібридне позиціонування.

В цілому на основі даних [3] можна скласти порівняльну таблицю – табл. 1.1, де можна продивитися параметри активних сервісів Wi-Fi позиціонування.

Табл. 1.1. Характеристики активних систем Wi-Fi позиціонування

Назва	**Унікальні мережі Wi-Fi**	**Реєстрацій пристроїв**	**Безкоштовне завантаження бази даних**	**Пошук SSID**	**Пошук BSSID**	**Ліцензія даних**	**Коментар**
Combain Positioning Service	> 2,4 млрд	> 67 млрд.	ні	так	так	Власна	Також база даних Cell ID.

Назва	Унікальні мережі Wi-Fi	Реєстрацій пристроїв	Безкоштовне завантаження бази даних	Пошук SSID	Пошук BSSID	Ліцензія даних	Коментар
Location API.org від Unwired Labs	> 1,5 млрд	> 4 млрд.	ні	так	так	Власна	Також база даних Cell ID
Служба локації Mozilla	> 1,287 млрд	> 104 млрд.	ні	ні	ні	Власна	Також база даних Cell ID, дані якої є у загальному доступі
Mylnikov GEO	860 млн.		так	ні	так	MIT	Також база даних ідентифікаторів стільників
Navizon	480 млн.	21,5 млрд.	ні	ні	так	Власна	На основі даних про натовп. Також база даних Cell ID.
radiocells.org	13 млн.		так	ні	так	ODbL	На основі даних про натовп. Також база даних Cell ID. У тому числі необроблені дані
OpenWLAN Map / openwifi.su	22 млн.		так	ні	так	ODbL	
WiGLE	506 млн.	7,2 млрд.	ні	так	так	Власна	Також база даних Cell ID.

Із наведеної таблиці видно, що переважна більшість подібних систем поширюються під пропрієтарними ліцензіями, тобто їх використання є платним. У таблиці наведено всього лише 3 продукти із відкритим доступом

до своїх баз даних, однак, два з них (що поширюються під ліцензією ODbL) зберігають інформацію про 13 та 22 млн. мереж, що складає менше 1 % від показника одного з лідерів - Combain Positioning Service.

Єдиною системою Wi-Fi позиціонування із відкритим доступом до власних баз даних і такою, що має їх конкурентоспроможне наповнення, виступає Mylnikov GEO – рис. 1.4.

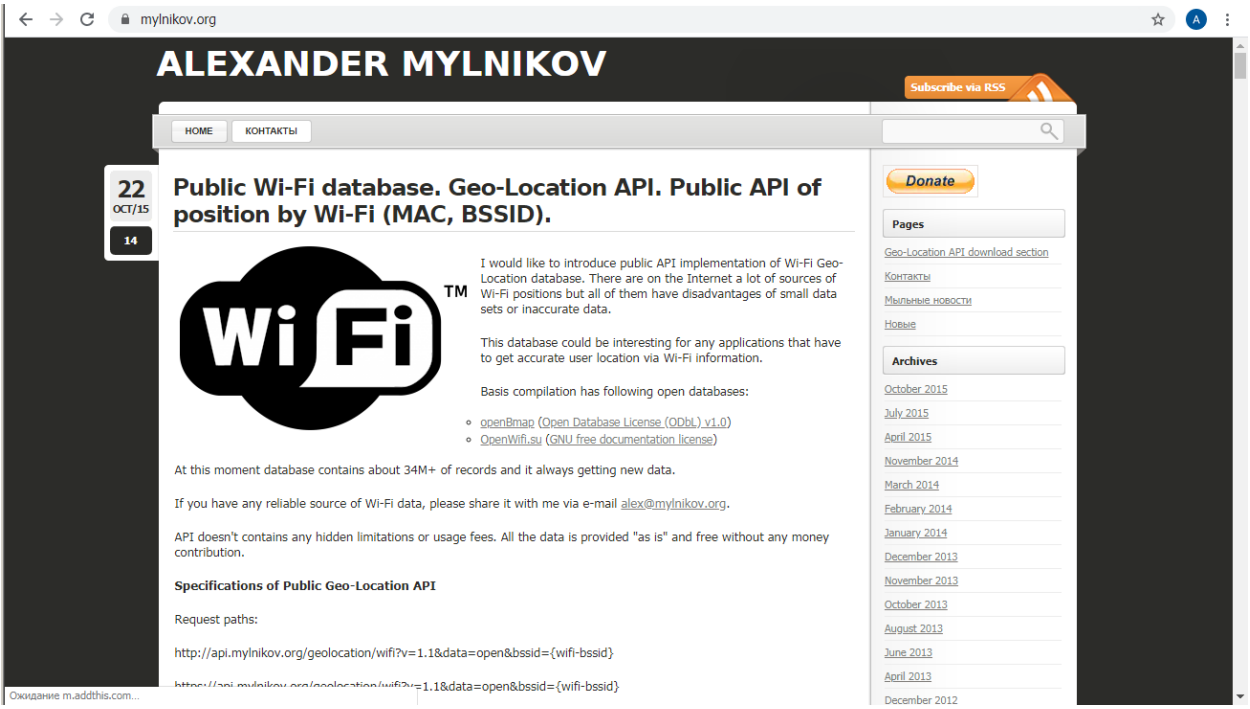


Рис. 1.4 Титульна сторінка системи WPS Mylnikov GEO

Навіть титульна сторінка цієї системи (на відміну від розглянутих вище аналогів) дивує надзвичайно простим дизайном, і, при ближчому розгляді виявляється, що даний, ніби дуже масштабний (на 860 млн. контрольованих мереж) продукт є персональною розробкою однієї людини – російського програміста Олександра Мильникова. Аналіз засобів, що тут надаються, співпадає зі стилем сайту і говорить про їх мінімалістичність. Нажаль, при урахуванні усіх факторів (надзвичайна простота, навіть скудність, системи та управління нею однією людиною – з одного боку, та декларовані величезні масштаби розробки – з іншого) неможливо із повною впевненістю відноситися до цього проекту.

Ще однієї негативною особливістю переважної більшості систем проведення Wi-Fi-позиціонування є закритість програмного коду, що реалізує дану процедуру. Відповідно, у широкої спільноти IT-спеціалістів зазвичай немає можливості перевірити роботу таких систем та впевнитися у достовірності даних, що вони надають. Зважаючи на те, що ці системи не дають широкому загалу конкретики, а лише надають масиви статистичних даних, у будь-якої особи, що звикла перевіряти вхідну інформацію, можуть виникнути цілком обґрунтовані питання про достовірність цієї інформації, її відповідність реальному стану речей.

1.3 Формулювання вимог

Таким чином, можна констатувати, що доцільною є розробка власного програмного забезпечення для моніторингу клієнтів Wi-Fi мереж.

Окреслимо вимоги до системи, що проектується:

- система моніторингу має будуватися на основі відкритих технологій, що дає можливість саму систему зробити відкритою, причому, як результати моніторингу (тобто має бути відкритим доступ до баз даних системи), так і вихідний код самої системи (для можливості перевірки адекватності її роботи усіма бажаючими сторонніми спеціалістами);
- система має бути розподіленою із центральним сховищем-сервером та клієнтами, що збирають інформацію та пересилають на централізований сервер;
- у системі має підтримуватися «живий» режим передачі зібраних даних (тобто коли клієнт отримує дані, одразу надсилає їх на сервер), однак при тимчасовій відсутності доступу до сервера слід передбачити можливість локального зберігання даних і відправленні їх пакетом при появі зв’язку з сервером;

- у системі мають бути застосовані надійні, зручні та продуктивні сховища даних, ймовірно, на основі якоїсь відкритої СУБД, обґрунтування вибору якої також слід провести;
- програмна складова системи має створюватися з використанням сучасних, популярних та функціональних мов програмування та інших засобів розробки (інтегровані середовища розробки, засоби адміністрування, продукти для захисту інформації, і т.п.).

Беручи до уваги ці відомості, можна переходити до розробки проектних рішень при створенні системи моніторингу клієнтів Wi-Fi мереж.

ВИСНОВОК ДО РОЗДІЛУ 1

В першому розділі була доведена доцільність мети дипломного проєкту, а також обґрунтована її актуальність. Були розглянуті та проаналізовані комерційні аналоги системи, та сформульовані вимоги до її реалізації. Були перераховані їх переваги та недоліки, завдяки цьому було скориговано вимоги до функціональності системи.

РОЗДІЛ 2.
ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДІВ, ЗАСОБІВ ТА ТЕХНОЛОГІЙ ДЛЯ ВИКОНАННЯ РОБОТИ

2.1 Обґрунтування архітектури системи

Для розробки архітектури такого комплексного рішення, як система моніторингу Wi-Fi клієнтів, спочатку слід докладно проаналізувати суть самої технології Wi-Fi та її особливості, що є важливими для даної предметної галузі.

Поняття бездротових технологій передачі даних є надзвичайно широким і, в першу чергу, може класифікуватися по просторовим масштабам відповідної системи передачі даних (рис. 2.1). Очевидно, що для кожного типу бездротових мереж існує певна специфіка процесу захисту інформації, тому такі типи слід розглянути докладніше.

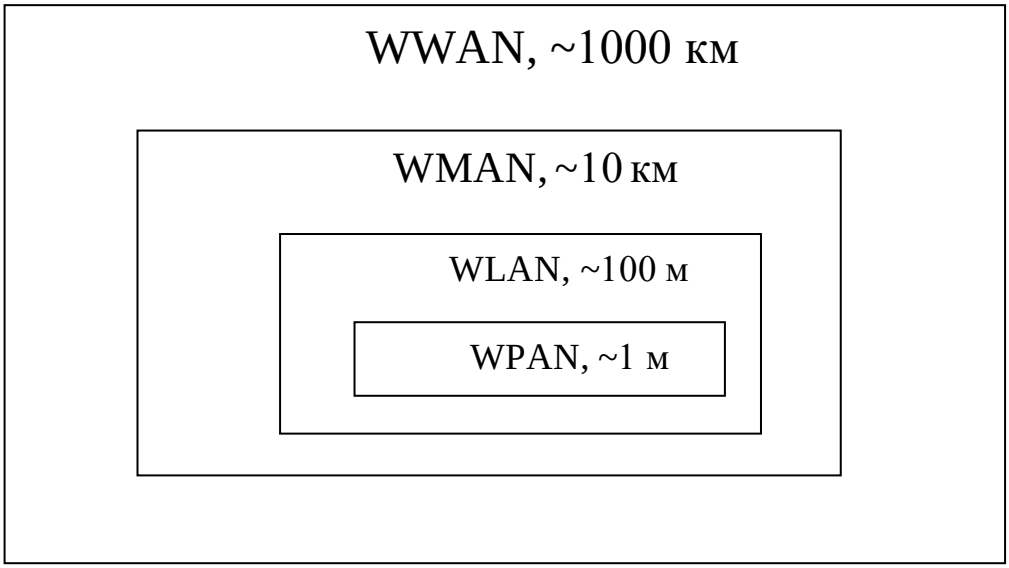


Рис. 2.1. Взаємне положення технологій бездротового доступу, залежно від географічних масштабів їх застосування

Найменші розміри мають персональні бездротові мережі WPAN (Wireless Personal Area Network), і прикладами таких технологій є Bluetooth та IrDA, що мають радіус дії порядку кількох метрів. Ці мережі мають непостійний, епізодичний характер і створюються самими користувачами за допомогою простого, неспеціалізованого обладнання (наприклад, засобами, що вбудовані у кожний мобільний телефон).

Мережі більшого розміру мають умовну назву WLAN (Wireless Local Area Network), і тут мова йде про масштаби до сотні метрів; прикладом відповідної технології є Wi-Fi. За допомогою Wi-Fi будуються локальні мережі підприємств, будинків, здійснюється організація доступу до мережі Інтернет у зоні «останньої милі». Технологія набула надзвичайного поширення і в межах міської забудови важко, практично неможливо знайти місце, де були б відсутніми локальні мережі Wi-Fi (а зазвичай таких мереж одночасно спостерігається до десятка і навіть більше).

Існують і бездротові мережі більших розмірів, які, зокрема, відносять до типу WMAN (Wireless Metropolitan Area Network), прикладом яких є мережі на основі протоколів WiMAX. Це системи масштабу міста, як витікає з їхньої назви, і вони поки що не отримали повсюдного впровадження (існує думка, що технології цього рівня не отримують широкого розвитку через конкуренцію збоку технологій, які описані у наступному пункті.

Найбільші масштаби (порядку цілої країни) мають бездротові мережі типу WWAN (Wireless Wide Area Network), як, наприклад, на основі технологій GPRS, LTE, і т.д. Часто вони базуються на існуючих мережах мобільного зв’язку, що знаходяться у власності національних мобільних операторів. Відповідна дорога інфраструктура, яка початково зводилася для організації голосового мобільного зв’язку, використовується також тепер і для передачі цифрових даних. Однак, швидкість передачі даних та порівняно висока вартість заважає цим технологіям «перемогти» Wi-Fi: він все ширше

впроваджується і залишається чи не найпопулярнішим способом організації локальних мереж.

Таким чином, у сучасному світі комп'ютерних інформаційних технологій під бездротовими технологіями практично у 99% випадків будь-хто матиме на увазі стек технологій Wi-Fi. Під цією назвою об'єднуються різноманітні стандарти, що усі разом мають код 802.11 американського Інституту інженерії електротехніки та електроніки (IEEE 802.11). Сама назва стандарту походить від словосполучення **Wi**reless **Fi**delity, що означає «бездротова вірність». З самого початку усі різновиди стандартів 802.11 розроблялися для використання у вільних від ліцензування радіодіапазонах. Всього існує більше 30 різновидів стандарту 802.11, але лише мала частка з них дійсно отримали широке впровадження у конкретних апаратних засобах.

В першу чергу, значний період часу практично усі пристрої Wi-Fi підтримували стандарт 802.11b. Він вийшов у 1999 році і найголовнішим удосконаленням цього стандарту (у порівнянні з початковою версією 802.11, 1997 р.) був перехід до несучих частот в діапазоні біля 5,5 ГГц, що зокрема дозволило підвищити швидкість передачі даних до 11 Мбіт/с (було 1 або 2 Мбіт/с при частоті біля 2,4 ГГц). Тривалий час цей стандарт займав лідируюче місце у апаратному забезпеченні різноманітних виробників.

Перед викладенням подальшого матеріалу слід відмітити, що існують певні особливості вимірювання швидкості передачі даних по каналам зв'язку, в т.ч. і бездротовим. По-перше, частіше всього швидкість оцінюють у мегабітах за секунду (Мбіт/с), на відміну, наприклад, від швидкості передачі інформації між окремими компонентами всередині ПК, яку «чесно» часто вимірюють у мегабайтах за секунду (Мбайт/с). Такий підхід, ймовірніше за все, обумовлений маркетинговою політикою, адже для кінцевого споживача, традиційно, чим більшою є швидкість, тим привабливіше продукт, а швидкість у мегабітах за секунду, звичайно, у 8 разів більша за величину тієї ж самої швидкості у мегабайтах за секунду. Цей нюанс завжди треба мати на увазі, так

як більшість програм, пов’язаних із завантаженням інформації з Інтернету, показують поточну (чи середню) швидкість у Мбайт/с, а кінцеві користувачі при цьому можуть надавати скарги, так як вважають, що система неправильно працює, адже швидкість передачі майже в 10 разів менше, заявленої швидкості роботи Wi-Fi обладнання. Насправді, швидкість буде меншою приблизно у 8 разів (1 байт = 8 біт), і це нормальна ситуація, зважаючи на різну одиниці виміру швидкості.

Ще один нюанс, пов’язаний із швидкістю передачі даних, полягає у тому, що корисна швидкість, яку бачитиме кінцевий користувач у своїх прикладних програмах (наприклад, менеджерах завантажень, клієнтах пірингових мереж, тощо) буде завжди меншою, ніж швидкість, заявлена у стандарті. Ця природна ситуація пояснюється тим, що анонсована швидкість є брутто-швидкістю (або, по-іншому, канальною) і з нею передаються усі дані по бездротовому каналу. Серед усіх даних є дані користувача (відповідно до задіяного протоколу прикладного рівня моделі OSI), а ще – значна кількість службових даних, що необхідні для роботи протоколів рівнів нижче прикладного (як мінімум, мережного IP та транспортного TCP). За деякими оцінками службові дані можуть досягати до 50% усієї інформації, що передається по бездротовому каналу, і це є ще однією причиною невідповідності реальної швидкості роботи користувача у мережі та швидкості стандарту. У подальшій роботі за умовчанням під швидкістю передачі даних будемо мати на увазі брутто-швидкість, а в інших випадках будемо наголошувати це окремо.

Отже, наступний «популярний» стандарт 802.11g з’явився у 2003 році і забезпечував швидкість до 54 Мбіт/с при частотах біля 5 ГГц (цей та усі наступні стандарти відповідали умові оберненої сумісності, тобто старе обладнання стандартів 802.11b і нижче могло працювати також і з новим обладнанням, що відповідало 802.11g, але на зниженій швидкості). Підвищення швидкості тут у великій мірі вдалося досягти за рахунок зміни

системи кодування сигналу (перехід до менш завадостійкого, тобто не настільки збиткового коду), а також ряду заходів радіотехнічного характеру. Цей стандарт використовується по даний час як основа для сучасних Wi-Fi мереж.

У 2009 р. з'явився стандарт 802.11n, який став гарним розвитком вищевказаних протоколів і розрахований на швидкості передачі даних до 600 Мбіт/с. Однак, суттєвим недоліком цього стандарту є необхідність наявності чотирьох (!) окремих антен у кожного пристрою, що працює в мережі, причому ще і відстань між цими антенами не може бути порядку сантиметру, а має дорівнювати хоча б десяткам сантиметрів.

Найсучасніші стандарти Wi-Fi приділяють надзвичайну увагу антенам, якими оснащений роутер. Їх робота може характеризуватися значною кількістю параметрів: кут антени, коефіцієнт підсилення і т.д. Далеко не завжди виробники антен докладно вказують ці дані. Із «антенних» характеристик роутерів завжди вказують режим MIMO (іноді під назвою «схема MIMO»). Розберемо докладніше, що мається на увазі під цією абревіатурою.

MIMO (в перекладі з англ. «Множинний вхід-вихід») - це технологія кодування сигналу, що дозволяє підвищити швидкість одночасної передачі даних на кілька пристроїв при використанні системи з декількох антен. Якщо в характеристиках роутера зазначено, що пристрій працює за схемою MIMO 2 × 2, то це означає, що дві антени приймають сигнал і дві передають. При цьому максимально можлива швидкість обміну буде вище, ніж при використанні однієї передавальної і однієї приймаючої антени, але необов'язково в два рази. Швидкість залежить ще і від кількості просторових потоків (spatial streams), які виникають при передачі даних в режимі MIMO. До того ж, запис «MIMO 2 × 2» не завжди означає, що у роутера саме дві антени, що працюють на прийом і передачу. У деяких випадках їх може бути три: одна приймає сигнал, друга

передає, а третя автоматично перемикається в різні режими роботи за потребою.

Існують також пристрої, що працюють за схемою MIMO 1 × 2, MIMO 3 × 3 і т.п. Чим більше приймальних антен має роутер, тим вище буде чутливість точки доступу в більшості випадків. Але слід пам'ятати, що сама по собі чутливість не гарантує високої швидкості передачі даних, так як багато що тут залежить від «засміченості» ефіру сторонніми сигналами використовуваного діапазону частот (сигнал на частоті 5 ГГц сильніше загасає при проходженні крізь стіни і меблі).

Останнім, на сьогоднішній день, важливим стандартом Wi-Fi можна назвати 802.11ac (2013 р.), що може забезпечувати швидкість передачі даних до 6,77 Гбіт/с, але при використанні аж 8 антен, що поки що мало представлено в реальних умовах локальних бездротових мереж України.

Вказані вимоги до антенних масивів сильно гальмують широке впровадження нових «швидких» стандартів, особливо, якщо взяти до уваги ступінь мініатюризації сучасних персональних «гаджетів», які на відміну від ноутбуків (та, тим більше, стаціонарних комп'ютерів, для яких взагалі бездротове підключення Wi-Fi все ще є, і, можливо, завжди залишатиметься переважно екзотикою) усе ширше використовуються для доступу до ресурсів мережі Інтернет через локальні мережі на базі Wi-Fi. Відповідно, найбільш поширеним стандартом на сьогоднішній день у сегменті SOHO (Small Office, Home Office) все ще залишається 802.11g.

Отже, розглянувши дуже докладно особливості самої технології Wi-Fi можна переходити до опису архітектури програмно-апаратного комплексу, що уособлює систему моніторингу клієнтів мереж Wi-Fi.

Рішення по моніторингу Wi-Fi мереж очевидно має складатися із численної кількості пристроїв-слухачів, що збирають інформацію, фізично знаходячись у різних місцях простору, який покривається контрольованою мережею. Усі ці пристрої в подальшому будемо називати клієнтами системи

моніторингу (тобто клієнтами проектованої системи), в той час як мобільні пристрої користувачів, що підключаються до мережі із цілями роботи в Інтернет, будемо називати клієнтами мережі Wi-Fi. Отже, усі клієнти системи моніторингу мають бути підключені до центрального вузла-сервера, що зберігатиме зібрану ними інформацію про клієнтів мережі Wi-Fi на довготривалій основі – рис. 2.2.

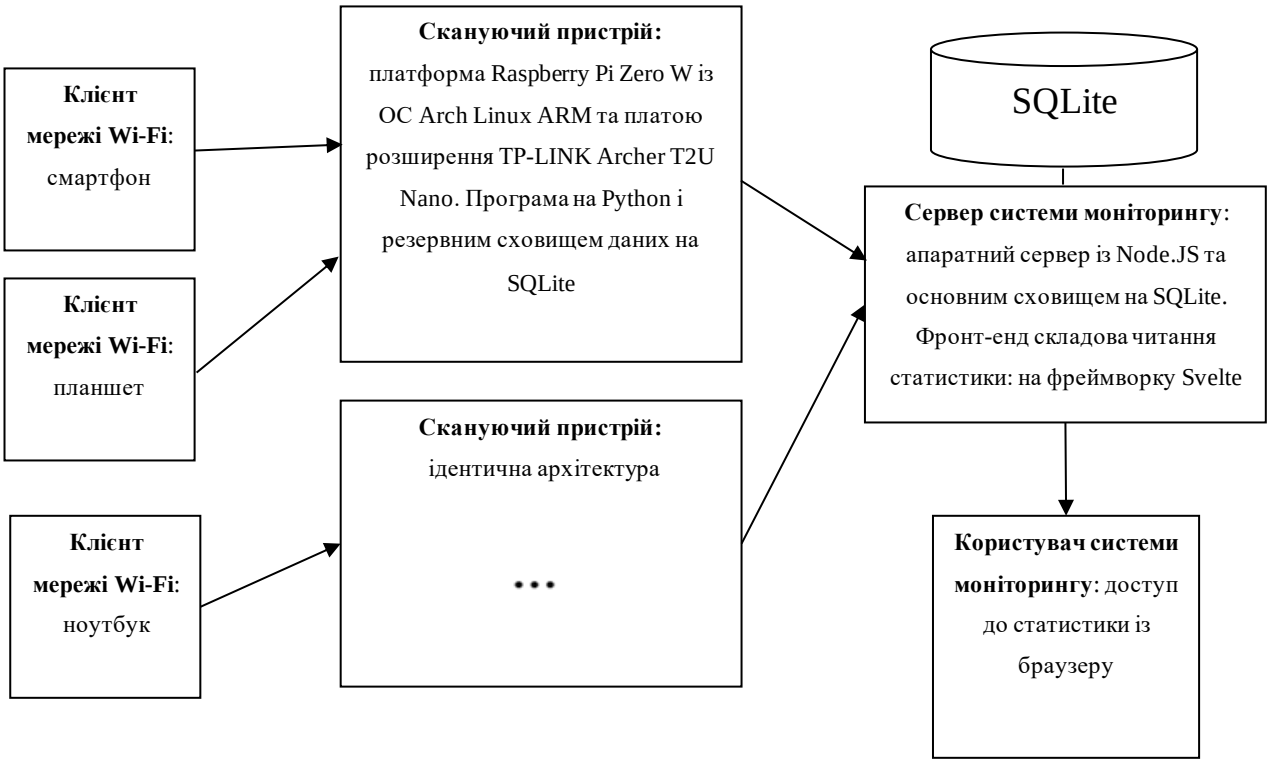


Рис. 2.2 Архітектура системи моніторингу клієнтів Wi-Fi мереж

2.2 Вибір засобів та технологій для обміну даними між складовими системи

Основними питаннями щодо обміну даними є вибір протоколу обміну даними та формату обміну між складовими системи.

У якості протоколу взаємодії між клієнтами системи моніторингу та сервером обрано HTTP.

HTTP - це протокол, що дозволяє отримувати різні ресурси, наприклад HTML-документи. Протокол HTTP лежить в основі обміну даними в Інтернеті.

HTTP є протоколом клієнт-серверного взаємодії, що означає ініціювання запитів до сервера самим одержувачем, зазвичай веб-браузером. Отриманий підсумковий документ може складатися з різних піддокументів, які є частиною підсумкового документа: наприклад, з окремо отриманого тексту, опису структури документа, зображень, відео-файлів, скриптів і багато чого іншого.

Клієнти та сервери взаємодіють, обмінюючись поодинокими повідомленнями, а не потоком даних. Повідомлення, відправлені клієнтом, зазвичай веб-браузером, називаються запитами, а повідомлення, відправлені сервером, називаються відповідями [4].

Хоча HTTP був розроблений ще на початку 1990-х років, за рахунок своєї розширюваності надалі він весь час удосконалювався. HTTP є протоколом прикладного рівня, який найчастіше використовує можливості іншого протоколу - TCP (або TLS – протокол над TCP, що використовує криптографічне шифрування) - для пересилання своїх повідомлень, проте будь-який інший надійний транспортний протокол теоретично може бути використаний для доставки таких повідомлень. Завдяки своїй розширюваності, він використовується не тільки для отримання клієнтом гіпертекстових документів, зображень і відео, але і для передачі вмісту серверів, наприклад, за допомогою HTML-форм. HTTP також може бути використаний для отримання тільки частин документа з метою поновлення веб-сторінки за запитом (наприклад за допомогою AJAX запиту). Саме зважаючи на ці особливості, HTTP і був обраний у якості основного протоколу передачі даних у системі.

Для передачі даних між клієнтами системи моніторингу та сервером вирішено використовувати формат JSON.

JSON - це абревіатура для JavaScript Object Notation, полегшеного формату, спочатку призначеного для обміну даними в Інтернеті. Вважається

підмножиною синтаксичного позначення, що представляє об'єкти, масиви, рядки, булеві числа та числа JavaScript.

Його популяризація відбулася близько 2001 року завдяки безумовній підтримці Дугласа Крокфорда. Yahoo! значно допомогло в його розповсюдженні після включення цього формату в деякі найбільш інноваційні веб-сервіси. У грудні 2006 року Google почав пропонувати свої канали JSON для свого веб-протоколу GData.

Незважаючи на те, що JSON заснований на ситаксисі JavaScript, він вважається окремою мовою формату даних, специфікація якої описана в RFC4627.

Анатомія розмітки JSON майже ідентична об'єкту JavaScript. Давайте поглянемо на приклад на рис. 2.3

```
{
 "id" : "0001",
 "type" : "donut",
 "name" : "Cake",
 "image" : {
    "url" : "images/0001.jpg",
    "width" : 200,
    "height" : 200
 },
 "dateEntry" : "2010-12-05"
}
```

Рис. 2.3 Приклад JSON об’єкту

Як ми бачимо, цей приклад практично ідентичний визначенню об’єкта в JavaScript: всі дані об’єкту розміщено між двома дужками, в межах яких параметри визначаються за парною схемою імені-значення. Кожен із термінів незмінно розмежований лапками - те, що в JavaScript не завжди потрібно, хоча й рекомендується.

Деякі особливості або правила формату JSON, які слід враховувати:

- Пари імен-значень завжди розмежовані лапками, незалежно від того, чи є вони дійсними іменами у JavaScript, де вони можуть з'являтися без них;
- JSON може представляти шість типів значень: об'єкти, масиви, числа, рядки, булеві та null;
- Дати не є окремим типом об'єкту;
- Числа в JSON не можуть мати нулі перед початком, за винятком числа 0 та дробових записів менше 1.

Масиви можуть використовуватися для збереження великої кількості групованих даних. За допомогою вкладених масивів і об'єктів можна створювати складну ієрархію даних та їх взаємодію. Нагадаємо, що у JavaScript поняття масиву і об’єкту дуже близькі, хоча це й не ідентичні сутності.

2.3 Обґрунтування вибору мов програмування

На клієнті системи моніторингу будемо використовувати мову Python (читають як «пайтон» або «пітон»). Це – відносно молода мова програмування загального призначення. Це означає, що писати цією мовою можна усе, що завгодно. Розробниками Python зазначається, що саме ядро мови є надзвичайно компактним (цим забезпечується висока швидкість програм, написаних на Python), але його супроводжують стандартні бібліотеки з широкими функціональними можливостями що це дозволяє використовувати дану мову для будь-яких цілей та вхідних даних [12].

Початково мова Python з’явилася у 1991 році, але у нашій країні довгий час була маловідомою (на відміну від популярних тоді C, Pascal, Basic). У великій мірі ця мова ставала все більше відомою за рахунок використання її у якості скриптової мови інтерфейсу CGI (Common Gate Interface, тобто механізму, який до появи PHP, ASP та інших вбудованих у тіло веб-сторінки засобів, забезпечував бізнес-логіку веб-сторінок, зокрема можливості доступу до баз даних).

Слід відмітити, що Python дозволяє створювати програми з використанням різних парадигм програмування: структурного, об’єктно-орієнтованого, функціонального, імперативного та аспектно-орієнтованого [13]. Звичайно, у більшості проектів використовується структурний (для порівняно малих програм) та об’єктно-орієнтований (ефективний для програм, що містять десятки тисяч рядків коду та більше) підхід до програмування.

У мови є офіційний сайт python.org, на якому можна завантажити декілька останніх стабільних версій компілятора. На даний момент доступною є версія 3.7.2, до якої включено порівняно просте, але достатньо наповнене середовище розробки IDLE (Integrated Development and Learning Environment). Загальний вид вікна цієї програми наведено на рис. 2.4. Наповнення цього середовища розробки можна назвати мінімалістичним, але достатнім для простих проектів (зокрема, для цілей навчання). При розробці професіональних проектів краще використовувати середовища із більшими функціональними можливостями, наприклад, Eclipse.

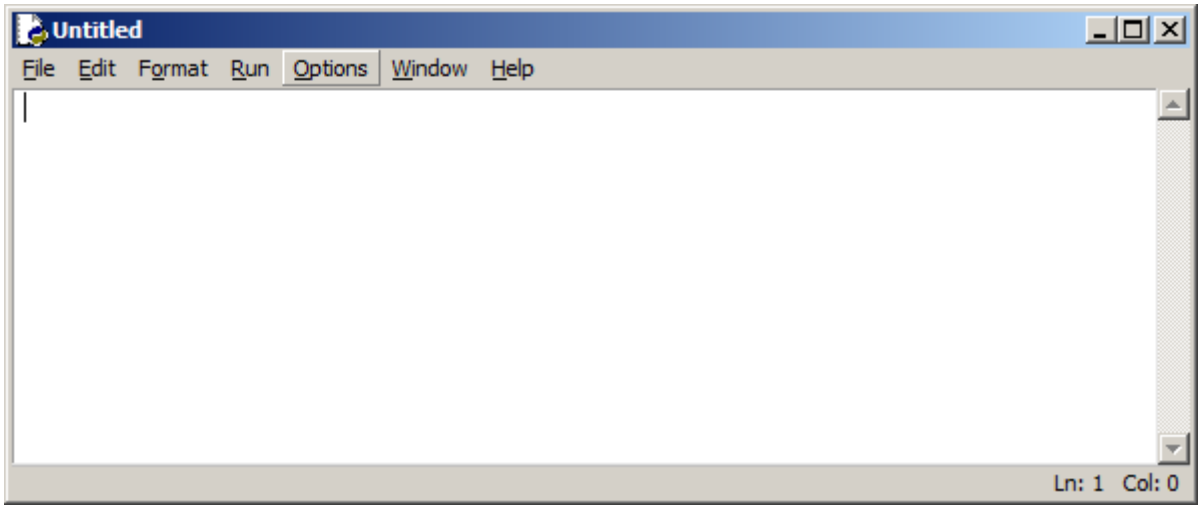


Рис. 2.4. Вікно стандартного середовища розробки IDLE, що поставляється разом з Python

Python був розроблений для легкого читання. Однією з його характеристик є використання слів, де інші мови використовували б символи. Наприклад, логічні оператори ‘!’, ‘||’ і ‘&&’ в Python будуть написані як ‘not’, ‘or’ і ‘and’, відповідно. Цікаво, що такі мови, як Паскаль та COBOL є одними з мов із дуже чітким синтаксисом, і обидві - з 70-х рр. Ідея чіткого та читабельного коду - це не щось нове [14].

Вміст блоків коду (циклів, функцій, класів тощо) перед кожним командним рядком, що належить до блоку, розмежовано пробілами або вкладками, відомими як відступи. Python, таким чином, відрізняється від інших мов програмування, які підтримують звичай оголошення блоків за допомогою набору символів, як правило, фігурними дужками {}. Можна використовувати як пробіли, так і символи табуляції для відступів коду, але не рекомендується їх змішувати.

Через синтаксичне значення відступу кожна інструкція повинна міститися в одному рядку. Однак, якщо ви хочете розділити інструкцію на кілька рядків для читабельності, додавши зворотну косу риску \ в кінці рядка, вказується, що інструкція продовжується в наступному рядку.

Коментарі можуть бути розміщені у два способи. Перший і найбільш підходящий для довгих коментарів - це використання позначень """коментар""", три лапки, що відкриваються, і три, що закриваються. У другому варіанті використовується символ ‘#’, такі коментарі продовжуються до кінця рядка.

Інтерпретатор ігнорує коментарі, що корисно, якщо ми хочемо вмістити додаткову інформацію для розробника в код, наприклад, пояснення поведінки частини програми.

Змінні визначаються динамічно, що означає , що ви не повинні вказувати їх тип заздалегідь і вони можуть приймати різні значення у різний час, навіть іншого типу відносно до одного, який був раніше. Для створення змінних використовується символ ‘=’.

Умовний оператор виконує свій внутрішній блок коду, тільки якщо виконується певна умова. Він визначається за допомогою ключового слова, ‘if’, за яким слідує умова та блок коду. Додаткові умови, якщо такі є, вводяться за допомогою ‘elif’ наступних умов та блоку коду. Усі умови оцінюються послідовно, поки не буде знайдено першу, що відповідає дійсності, і пов'язаний з нею блок коду є єдиним, що виконується. За бажанням може бути вказаний блок без умови (ключове слово, ‘else’, за яким кодовий блок), який буде виконано лише тоді, коли всі умови були помилковими.

Цикл for схожий на foreach в інших мовах. Проводиться цикл через ітерований об’єкт, такий як список, кортеж або генератор, і для кожного елемента ітерованого об’єкту він виконує внутрішній блок коду. Він визначається за ключовим словом, ‘for’ за яким йде ім'я змінної, потім ключове слово ‘in’, потім – ітерований об’єкт, і, нарешті, внутрішній блок коду. На кожній ітерації зазначеному імені змінної присвоюється наступний елемент ітерованого об’єкту.

Для декларування списків використовуються дужки ‘[’ та ‘]’, а для декларації кортежів використовуються дужки ‘(’ та ‘)’. Елементи розділені

комами, а у випадку кортежів необхідно, щоб була принаймні одна кому, аби відрізнити кортеж від простих дужок.

І списки, і кортежі можуть містити елементи різних типів. Однак списки зазвичай використовуються для елементів одного типу в змінній кількості, тоді як кортежі зарезервовані для різних елементів у фіксованій кількості.

Для доступу до елементів списку або кортежу використовується індекс (починаючи з "0", а не "1"). Негативні індекси можна використовувати для доступу до елементів з кінця [15].

Списки характеризуються тим, що вони змінні, тобто, ви можете змінювати їх зміст під час виконання, в той час як кортежі незмінні, тому що це їх зміна після створення не представляється можливою.

Для оголошення словника використовуються дужки ‘{’ та ‘}’. Вони містять елементи, розділені комами, де кожен елемент складається з пари ключ-об’єкт, а символ ‘:’ відокремлює ключ від відповідного значення.

Значення в словниках можуть бути змінними, тобто, ви можете змінити їх значення під час виконання. Натомість ключі словника повинні бути незмінними. Це означає, наприклад, що ми не зможемо використовувати списки або інші словники, як ключі [16].

Значення, пов’язане з ключом може буди будь-якого типу даних, навіть іншим словником.

Функції визначаються за допомогою ключового слова ‘def’, за яким слідує ім'я функції і її параметри. Ще один спосіб написання функцій, хоча і менш використовуваний - за допомогою ключового слова ‘lambda’, яке з'являється у функціональних мовах, таких як Lisp [17].

Значенням, що повертається функціями, що були визначені за допомогою ключового слова ‘def’ буде значення, задається інструкцією ‘return’.

Ці класи визначаються за допомогою ключового слова ‘class’, за яким слідує ім'я класу і, якщо клас успадковується від іншого класу, ім’я цього класу.

У Python 2.x було рекомендовано, щоб усі класи успадковували вбудований клас ‘Object’, у Python 3.x це вже не потрібно[18].

"__init__" - це спеціальний метод, який виконується при створювані екземпляру класу, він, як правило, використовується для ініціалізації атрибутів та виконання необхідних методів. Як і всі методи в Python, він повинен мати принаймні один параметр, що зазвичай називається ‘self’. Решта параметрів будуть тими, що мають вказуватися при створювані екземпляру класу.

Атрибути, до яких потрібно отримати доступ поза межами класу, повинні бути оголошені, використовуючи ‘self.’ перед їх іменами.

В Python немає поняття інкапсуляції, тому програміст повинен нести відповідальність за привласнення значень атрибутів.

Існує багато можливостей, які можна додати до мови, імпортуючи модулі, більшість також написана на Python, які забезпечують певні функції та класи для виконання певних завдань[19]. Прикладом може слугувати модуль Tkinter, який дозволяє створювати графічні інтерфейси на базі бібліотеки Tk. Інший приклад - модуль os, який забезпечує доступ до багатьох функцій операційної системи. Модулі додаються до кодів шляхом написання ключового слова ‘import’, а потім імені модуля, який ми хочемо використовувати.

У Python є велика стандартна бібліотека, яка використовується для різних завдань[20]. Це походить від філософії "батареї включені" ("batteries included") модулів Python. Стандартні модулі бібліотеки можуть бути доповнені спеціальними модулями, написаними як на C, так і на Python. Завдяки широкому розмаїттю інструментів, що входять до стандартної бібліотеки, у поєднанні з можливістю використання мов низького рівня, таких

як C і C ++, які здатні взаємодіяти з іншими бібліотеками, Python - це мова, яка поєднує свій чіткий синтаксис з величезною силою менш елегантних мов.

2.4 Вибір платформи та мови програмування для реалізації серверної логіки системи.

На сервері вирішено використовувати систему Node.JS. NodeJS був придуманий Райаном Далем в 2009. Ще до створення NodeJS, Даль займався розробкою на Ruby on Rails. Основну ідею для NodeJS він запозичив з Flickr (сайт для завантаження та обміну зображеннями). На сайті була шкала стану, яка показувала статус завантаження зображення. Зараз це здається чимось самим собою зрозумілим, але тоді, на початку нульових, подібне «спілкування» з боку сервера було чимось новим і незвичайним. Саме це і «зачепило» Даля, адже раніше сервери не мали можливостей для одночасної обробки декількох запитів (наприклад, при завантаженні зображення і паралельних запитах з боку того ж користувача). Це явище називається паралелізмом [7].

Веб-серверам того періоду потрібно було навчитися одночасній обробці декількох запитів. Поточний метод отримання запиту / відправлення відповіді був явно застарілим і мало підходив для вирішення нових практичних завдань. Більш наочно вся логіка процесу зображена на графіку рис. 2.5.

Нехай, наприклад, виникає запит на завантаження файлу. Сервер приймає цей запит і поки він виконується (товста лінія праворуч), відбувається обробка інших запитів. Потім повертається відповідь на головний запит. Такою є концепція паралелізму з боку NodeJS[8].

Даль з колегами реалізували цю концепцію в JavaScript з використанням V8 Engine від Google. Цей движок бере JavaScript код і компілює його в C++, причому, відбувається це з блискавичною швидкістю. V8 виконує операції на льоту, і швидко компілює код перед його виконанням. Це робить NodeJS швидше за усі інші мови програмування та фреймворки.

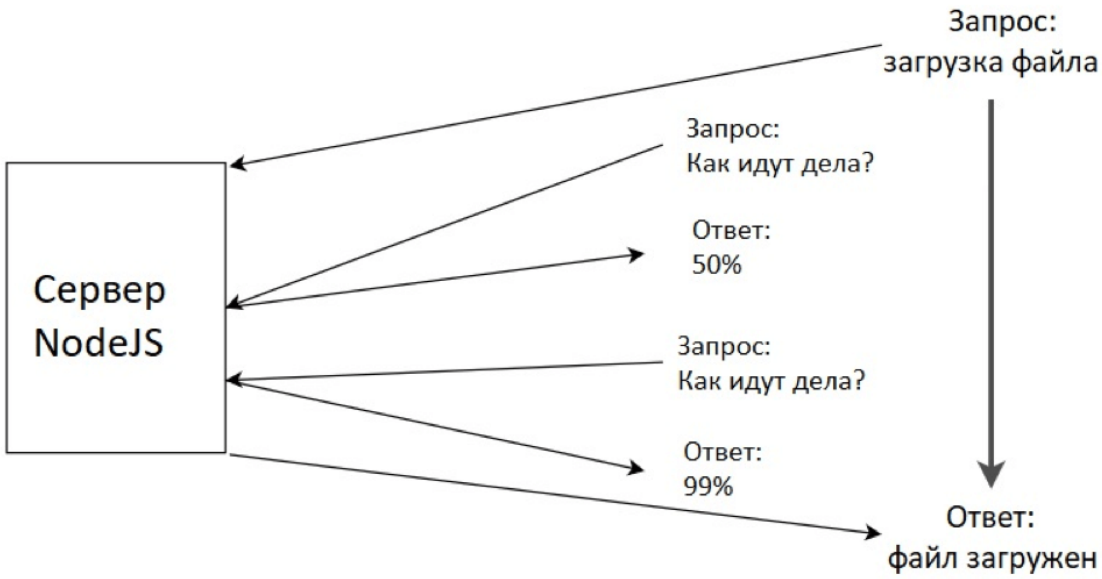


Рис. 2.5 Схема взаємодії із сервером на базі NodeJS

NodeJS не був стандартним веб-сервером ні раніше, ні зараз. При розпакуванні NodeJS він не запускається відразу як веб-сервер [9]. Для цих цілей розробникам будуть потрібні фреймворки або прописування логіки в NodeJS.

Чому ж NodeJS не є стандартним веб-сервером, якщо він був задуманий саме для цього? Вся справа в тому, що NodeJS створювався для вирішення більш широкого спектра завдань, ніж просто багатозадачність сайту. NodeJS замислювався для реалізації ідеї створення кількох запитів і одночасної обробки декількох процесів. В цьому і криється популярність платформи, адже NodeJS - це куди більше, ніж просто веб-сервер[10].

Окрім використання NodeJS у якості веб-серверу, із нього виходить відмінна операційна система, засіб логування (наприклад, для запису HTTP запитів або призначеного для користувача поведінки в десктопному додатку), універсальна скриптова мова. Ще одна цікава область застосування - створення нейронних мереж і машинне навчання. Проте, найчастіше NodeJS все ж таки використовується для зведення веб-серверів, що саме і виконано у даній роботі[11].

Відмітимо, що основною мовою програмування у системі є JavaScript.

JavaScript - це спеціалізована мова програмування, яка традиційно використовувалася для управління об'єктною моделлю документа у браузері під час перегляду сторінок мережі Інтернет[5].

Особливістю JavaScript є те, що його висхідні програмні коди інтерпретуються, що є досить гнучким, але повільним рішенням.

Оператори у цій, в цілому С-подібній мові, розділяються крапкою з комою. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами).

Однорядковий коментар виділяється символом //, а багаторядковий - парою символів / * і * /, в чому JavaScript знову повторює С.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням.

Мова JavaScript об'єктно-орієнтована, проте заснована на прототипах, а не на класах. Є чотири типи об'єктів: вбудовані об'єкти, об'єкти браузера, об'єкти документа і об'єкти користувача (програміста).

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За умовчанням передбачається, що доступ до локальної файлової системи заборонений. Однак браузери можуть надавати спеціальні об'єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку виконання файлових операцій.

Сценарії JavaScript активно взаємодіють з об’єктами, вбудованими в Web-сторінку. Для цього вони, власне, і створюються. Але перш, ніж ця взаємодія стане можливою, слід впровадити код сценарію в текст HTML-документа. Існує кілька способів зв'язати HTML-документ з конкретним сценарієм (скриптом), але зазвичай їх просто розміщують всередині контейнерного тегу <SCRIPT>, тобто між дескрипторами <script> і </script>.

Контейнер <SCRIPT> в цьому випадку буде перебувати безпосередньо в HTML-документі, причому у довільному його місці. Програмний код пишуть прямо в HTML-документі або в спеціальних текстових файлах, які можна викликати з головного HTML-документа. Для початку розглянемо перший варіант. Перш за все браузер знаходить тег <script> в тілі веб-документа, і весь наступний текст намагається обробити як скриптовий код. І так до тих пір, поки не зустріне закриваючий тег </script>. Після цього всі наступні символи будуть вважатися HTML-текстом. Будь-який HTML-документ може містити довільне число «скриптових включень», але кожне має відкриватися і завершуватися відповідним тегом. Від їх розташування у тілі HTML-документа іноді може залежати функціонування всієї Web-сторінки, але про це буде сказано пізніше.

Контейнерний тег <script> може містити атрибут SRC, який вказує ім’я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут необхідний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай використовують js.

Якщо сценарій розташовується в окремому файлі, то в ньому, зрозуміло, теги <SCRIPT> і </ SCRIPT> не пишуть. Сценарій, завантажений з зовнішнього файлу, можна уявити собі просто як його вставку в HTML-документ.

Як уже зазначалося, в окремих файлах зазвичай розміщують бібліотеки функцій (визначення функцій), а також сценарії, які використовуються в

декількох HTML-документах одного або декількох сайтів. Сценарій можна також писати у вигляді рядка операторів, між якими ставиться крапка з комою. Такий рядок, взятий в лапки, служить у якості значення атрибута-події.

Виклики функцій і їх визначення можуть слідувати в довільному порядку, але тільки якщо вони розташовані в одному і тому ж контейнері <script>. Якщо вони розміщуються у різних контейнерах <script>, то необхідно, щоб визначення функції передувало її виклику. Аналогічно, якщо визначення функцій знаходяться в окремому файлі, то його необхідно завантажити в HTML-документ раніше викликів цих функцій.

При спробі завантажити і виконати неправильний варіант HTML-коду з'явиться діалогове вікно з повідомленням «Помилка: Передбачається наявність об’єкта». Браузер інтерпретує теги HTML послідовно. Так, зустрівши вираз виклику функції myfunc() в одному контейнері <script>, браузер ще не має в пам'яті визначення цього об'єкта (функції), розташованого в іншому контейнері <script>. Якщо ж визначення функції і її виклик знаходяться в одному і тому ж контейнері <script>, то його вміст спочатку завантажується в пам'ять, а потім аналізується. Коли інтерпретатор зустрічає виклик функції, то він шукає її визначення в пам'яті і в разі успіху виконує код цієї функції.

Якщо необхідно, щоб сценарій виявився в браузері перш, ніж буде завантажено елементи HTML-документа, то його слід розташувати у верхній частині HTML-коду, а ще краще в контейнері <head> в заголовку документа. Такими є особливості браузерного варіанту використання мови JavaScript, що з’явився ще біля 25 років тому назад.

На сьогоднішній день JavaScript - це мультипарадигмова мова програмування, яка підтримує об'єктно-орієнтований, імперативний і функціональний стилі; є реалізацією стандарту ECMAScript (стандарт ECMA-262).

На сьогодні JavaScript використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить все ще у браузерах як мова сценаріїв для додання інтерактивності веб-сторінок, однак може застосовуватися і як серверна мова у складі розглянутого вище NodeJS, так і для розробки управляючих програм для деяких типів електронної техніки.

Основні архітектурні риси мови на сьогоднішній день - це динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції, як об'єкти першого класу. На JavaScript вплинули багато інших мов програмування, але початково при її розробці була мета зробити мову схожою на Java. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці. Саме зважаючи на ці та інші описані особливості даної мови, вона прийнята у якості основи у розробці, що виконується.

2.5 Вибір засобів відображення зведеної інформації для серверної частини системи.

Користувач системи моніторингу, що проектується, звичайно повинен мати можливість підключитися до неї та отримувати статистичну інформацію (для чого, власне, і задумувалася ця система) про клієнтів Wi-Fi мереж. Звичайно, краще робити це у зручному візуальному режимі, причому бажано – через браузер, тобто кросплатформенно.

Для цілей відображення результуючої інформації клієнту системи моніторингу розроблено спеціальне програмне рішення з використанням фреймворку Svelte.

Svelte - це вільний фреймворк для мови JavaScript із відкритим кодом, написаний Річем Харрісом. Програми Svelte не включають посилання на сам фреймворк, натомість, створення програми Svelte генерує код для маніпуляції з DOM, що може забезпечити кращу ефективність роботи клієнта [6]. У Svelte

є власний компілятор для перетворення коду програми в JavaScript на стороні клієнта під час створення, який написано в TypeScript. Вихідний код Svelte має ліцензію під ліцензією MIT і розміщується на GitHub.

2.6 Вибір засобу сховища даних

SQLite - це вбудована реляційна база даних, що поставляється з вихідними кодами. Вперше випущена в 2000 році, призначена для надання звичних можливостей реляційних баз даних без властивих їм накладних витрат. За час експлуатації встигла заслужити репутацію як мобільна, легка у використанні, компактна, продуктивна і надійна база даних.

Вбудовуваність бази даних означає, що вона існує не як процес, окремий від обслуговується процесу, а є його частиною – частиною деякого прикладного застосування. Зовнішній спостерігач не помітить, що прикладний додаток користується СУБД. Додаток просто робить його роботу, движок БД просто міститься всередині. Це позбавляє від необхідності мережевих налаштувань і адміністрування. Такий підхід дуже сильно полегшує життя програмісту: ніяких фаєрволів, ніяких мережних адрес, ніяких користувачів і конфліктів їх прав доступу. І клієнт, і сервер працюють в одному процесі і це позбавляє від проблем конфігурації, адже усе, чого потребує програміст, уже скомпільовано в його додатку.

Будь-яка кількість потоків може без проблем отримати доступ до однієї бази даних. Багато запитів для читання можуть виконуватися паралельно. Запити для запису можна виконувати лише за умови, що одночасно не відбувається інший такий запит. В такому випадку запит для запису не виконається, повертаючи код помилки, або він може автоматично спробувати ще раз, і так поки не закінчиться налаштована кількість спроб.

Бібліотека реалізує більшість стандартів SQL-92, включаючи транзакції атомарних баз даних, узгодженість бази даних, ізоляцію та довговічність (ACID), тригери та найскладніші можливі запити.

Python включає підтримку SQLite, створеної на базі версії 2.5, вбудованої в стандартну бібліотеку Python як модуль ‘sqlite3’.

Зважаючи на усі ці переваги, SQLite приймається за основу для сховищ даних (резервного – на клієнтах системи моніторингу та основного – на сервері) проектованої системи.

ВИСНОВОК ДО РОЗДІЛУ 2

В розділі були досліджені та описані технології, потрібні для створення клієнт-серверних систем, браузерних графічних інтерфейсів, та пристроїв для сканування та обробки Wi-Fi пакетів.

Були обрані мови програмування, фреймворки, бібліотеки та модулі. Для простоти розробки, як для сервера, так і для браузерного клієнта було обрано JavaScript. Для скануючих пристроїв було обрано Python, через його широку підтримку та існування бібліотеки scapy.

Для клієнтського інтерфейсу було обрано фреймворк Svelte через його простоту, зручність та швидкість. Серверним фреймворком було обрано NodeJS через використання JavaScript, а також його підтримка сучасних протоколів, таких як EventStream.

Бібліотека scapy була обрана як найкраща серед бібліотек для роботи з бездротовими інтерфейсами та отримання й обробки Wi-Fi пакетів.

Для автоматизації конфігурування системи, пошук в локальній мережі та під'єднання скануючих пристроїв відбувається за допомогою UDP-трансляції.

Також були використані такі стандартні веб-технології як SQLite, HTML, CSS, EventStream та інші.

Обрані технології дозволяють реалізувати поставлену мету – створення простої для розгортання системи збору статистики щодо пересування мас людей.

РОЗДІЛ 3.
РОЗРОБКА СИСТЕМИ

3.1 Визначення вимог і завдань

Необхідно створити програму для пристроїв-сканерів, що буде отримувати та зберігати MAC-адреси пристроїв, що були відскановані, а також передавати цю інформацію по запиту від координуючого сервера. Також потрібно створити програму для самого координуючого сервера, та його клієнтський інтерфейс.

Ця система має виконувати наступні функції:

- Можливість відображення списку підключених сканерів та їх статуси;
- Можливість отримання інформації щодо пристроїв, відсканованих кожним пристроєм;
- Можливість схематичного розташування пристроїв на віртуальній карті та моделювання напрямів пересування відсканованих пристроїв;
- Можливість отримання історичної інформації з попередніх пунктів.

3.2 Вибір мов програмування

Для розробки програмних продуктів було обрано мови програмування Python для програмування поведінки пристроїв та JavaScript для контролюючого сервера.

Вибір мови Python зумовлений такими причинами:

- це об'єктно-орієнтована мова програмування з динамічною типізацією, що дозволяє швидке прототипування та гнучкість розробки;

- має багату бібліотеку стандартних рішень, що подальше спрощує розробку;
- має надзвичайну портативність та підтримку майже будь-яких операційних систем та платформ.

Вибір JavaScript для контролюючого сервера та браузерного клієнта зумовлений такими причинами:

- це де-факто стандартна мова для клієнтських рішень, що найбільш широко підтримується всіма можливими браузерами;
- небажання використовувати ще одну мову для контролюючого сервера, та широкий вибір готових серверних рішень, зумовлений її популярністю поза межами браузерного середовища.

3.3 Вибір допоміжних компонентів програми

Для взаємодії з Wi-Fi інтерфейсом пристроїв використовуються:

- команди операційної системи Linux ‘ip’ та ‘iw’;
- модифікація драйвера бездротових мережевих інтерфейсів ‘aircrack-ng’;
- бібліотека ‘scapy’ для мови Python використовується для збору та обробки бездротових інформаційних пакетів.

Для зберігання, передачі та відображення інформації використовуються:

- база даних SQLite для зберігання даних;
- HTTP запити у форматах JSON та EventStream для їх передачі;
- та фреймворк Svelte з його компонентами для їх відображення.

3.4 Проєктування

Система розташована в одній локальній мережі, та складається з основного серверу, декількох скануючих пристроїв, та клієнта, що може під’єднуватися до сервера для отримання результатів роботи системи. Структурну схему системи подано у додатку 2.

Скануючий пристрій виконує такі ролі в системі:

- сканування та відстежування Wi-Fi пакетів у радіопросторі;
- зберігання відсканованих пакетів для подальшого аналізу;
- передача інформації на сервер в реальному часі.

Сервер виконує такі ролі в системі:

- отримання та зберігання інформації про відскановані пристрої;
- обробка цієї інформації;
- знаходження скануючих пристроїв в локальній мережі та під'єднання до них.

Клієнт виконує функцію відображення інформації, що була отримана та оброблена сервером, а саме:

- перелік доступних скануючих пристроїв;
- їх пошук та фільтрація;
- візуальне розташування скануючих пристроїв для аналізу пересування людських мас;
- візуалізація детальної інформації щодо відсканованих пристроїв, їх MAC-адрес, тощо.

3.5 Опис алгоритму системи

В системі реалізовано алгоритм, що складається з таких кроків:

- Запуск сервера;
- Запуск скануючих пристроїв;
- Знаходження пристроїв за допомогою UDP discovery;
- Встановлення HTTP з'єднання за протоколом EventStream;
- Сканування та передача даних в реальному часі від пристроїв до сервера;
- Запуск клієнта;
- Запит на сервер про підключені скануючі пристрої та інші дані;

- Відправка даних, що відповідають запиту, на клієнт;
- Отримання даних клієнтом;
- Візуалізація та відображення отриманих даних.

3.6 Вимоги до технічного забезпечення

Апаратне забезпечення:

- Серверний комп'ютер з 4-ядерним процесором, об'ємом оперативної пам'яті не менше ніж 1 Гбайт, та об'ємом вільного дискового простору не менше ніж 1 Гбайт, наявне устаткування для встановлення локальної мережі Wi-Fi;
- Комп'ютер, що має можливість відображення веб-сторінок та підключення до мережі Інтернет;
- Маленькі одноплатні комп'ютери OrangePi Zero, з 4-ядерним процесором ARM Cortex A6, та 256 Мбайт оперативної пам'яті, з можливістю під'єднання до локальної мережі за допомогою Wi-Fi та Ethernet;
- Модуль Wi-Fi TP-LINK Archer T2U Nano.

Програмне забезпечення:

- Windows 7/8/10 або Linux, Arch Linux ARM;
- Python 3.6;
- NodeJS 9;
- SQLite;
- Scapy;
- Intellij IDEA.

3.7 Вимоги до технічного забезпечення

Для налаштування скануючого пристрою потрібно послідовно виконати такі кроки:

- Налаштувати SD-картку - встановити на неї заготований образ з Arch Linux ARM та встановленою програмною частиною, потім долаштувати операційну систему на картці для підключення до локальної мережі;
- Зібрану платформу Orange Pi Zero потрібно встановити у бажаному місці;
- Далі треба встановити в неї налаштовану SD-картку;
- Встановити Wi-Fi модуль в USB-порт пристрою.
- Для налаштування серверу потрібно:
- Підключити його до локальної мережі;
- Встановити необхідну локальну адресу;
- Запустити програмну частину серверу.

Система не потребує подальшого налаштування, оскільки сканери автоматично будуть знайдені та підключені сервером.

Для роботи з клієнтською частиною необхідно лише підключитися до серверу з будь-якого браузера.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі були розглянуті та описані технічні вимоги для роботи системи, а також опис алгоритму її роботи та інструкція з налаштування та встановлення системи.

Так, за допомогою інструкції по налаштуванню, можна запровадити дану систему для використання у офісах, ТРЦ, підприємствах та будь-яких інших установах.

РОЗДІЛ 4.
ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ

4.1 Робота скануючого пристрою

Систему для сканування навколишніх Wi-Fi пристроїв було успішно розроблено та протестовано.

На рис. 4.1 зображено знаходження сервером двох скануючих пристроїв, що знаходяться у локальній мережі сервера, за допомогою команди ncat операційної системи Linux.

Рис. 4.1 Демонстрація автоматичного знаходження скануючих пристроїв у локальній мережі

На рис. 4.2 зображено передачу на сервер даних про MAC-адреси знайдених пристроїв та час їх знаходження в реальному часі за допомогою команди curl операційної системи Linux.

```
λ curl -s localhost:8000/listen
event: scan
id:0
data: {"ts":1591311273211,"mac":"a08869a25c0b"}

event: scan
id:1
data: {"ts":1591311273566,"mac":"a08869a25c0b"}

event: scan
id:2
data: {"ts":1591311273577,"mac":"a08869a25c0b"}
```

Рис. 4.2 Інформація що передається в реальному часі скануючим пристроєм

4.2 Робота сервера

Сервер приймає та обробляє дані, а також передає їх за запитом на клієнт. На рис. 4.3 відображена таблиця збережених даних, де перший стовпець це ідентифікатор пристрою, другий це час надходження пакету, а третій це MAC-адреса відсканованого пристрою.

	host	ts	mac
1	main	1584913790597	84c7ea88ff02
2	orangepi	1584913790613	84c7ea88ff02
3	orangepi	1584915814482	baf447961bb6
4	orangepi	1584915815433	a4db30b30fe7
5	main	1584915816002	26a3f1f44a0a
6	orangepi	1584915816024	26a3f1f44a0a
7	main	1584915851193	742f68d587cc
8	orangepi	1584915854981	742f68d587cc
9	main	1584916010349	daa11953c3e0
10	orangepi	1584916010398	daa11953c3e0
11	main	1584916010403	daa11953c3e0
12	main	1584916010408	daa11953c3e0
13	orangepi	1584916010414	daa11953c3e0

Рис. 4.3 Приклад даних, що зберігаються на сервері

4.3 Робота клієнта

Браузерний клієнт використовується для відображення результатів сканування. Робота з ним виконується в такому порядку:

- Підключення до серверу;
- Отримання списку скануючих пристроїв;
- Пошук бажаних пристроїв у списку;
- Розташування бажаних пристроїв на “мапі”;
- Отримання статистичної інформації від розташованих пристроїв.

На рис. 4.4 показано початковий стан списку пристроїв - повідомлення про те, що пристроїв не було знайдено.

Рис. 4.4 Список пристроїв за відсутності пристроїв

Пристрої будуть автоматично знайдені та отримані від сервера у режимі реального часу.

На рис. 4.5 та рис 4.6 показано список пристроїв зі знайденими пристроями, та продемонстровано роботу пошуку по списку. Для тестування системи з великою кількістю скануючих пристроїв, крім двох справжніх пристроїв (main, orangepi) також було штучно додано декілька імітуючих (test1, test2, ...).

Рис. 4.5 Список знайдених пристроїв

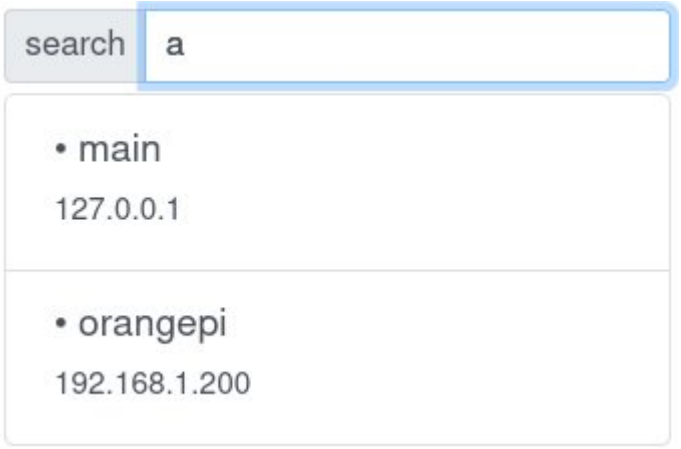


Рис. 4.6 Пошук серед знайдених пристроїв

Далі, пристрої перетягуються за допомогою Drag&Drop та розташовуються на основному робочому просторі, рис. 4.7.

На просторі можливо спостерігати статистику сканування пристроями MAC-адрес, з частотою їх появи, рис. 4.8

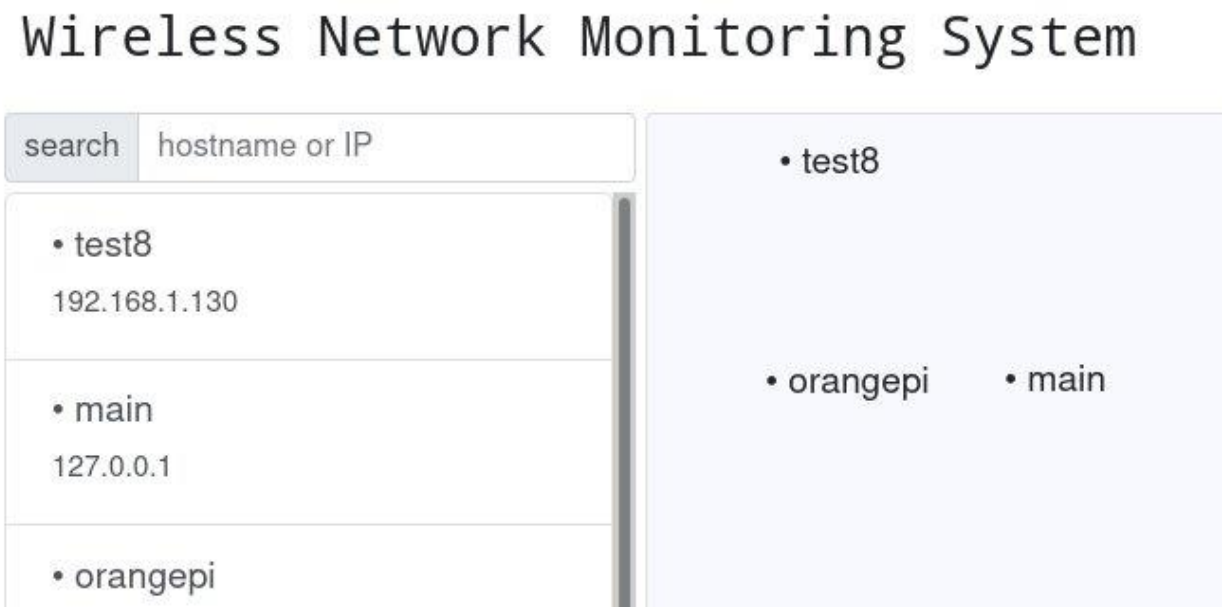


Рис. 4.7 Пристрої розташовані на робочому просторі

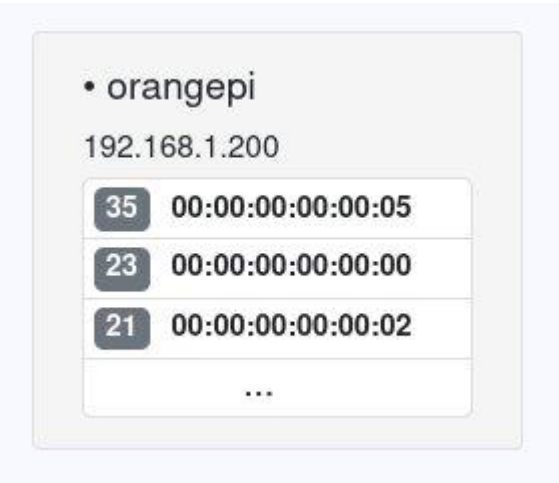


Рис. 4.8 MAC-адреси та частота їх появи, що отримані від скануючого пристрою

ВИСНОВОК ДО РОЗДІЛУ 4

В даному розділі були продемонстровані результати роботи системи сканування користувачів бездротових мереж Wi-Fi.

Так представлено результати виміру на скануючому пристрої, зберігання даних у серверній базі даних та демонстрація візуального представлення даних у браузері, що демонструє працездатність системи.

ВИСНОВКИ

Під час виконання дипломного проекту були досягнуті поставлені цілі, а саме:

- проведений аналіз існуючих аналогів;
- були сформульовані основні вимоги, на основі котрих розроблялась система;
- були обрані необхідні технології, що дозволяють системі виконувати поставлені задачі;
- та реалізована система.

Після завершення розробки система була перевірена на працездатність та протестована.

Можна зробити висновок, що маючи певний набір універсальних програмних модулів та компонентів, та навичок роботи з ними, можна створювати будь-які програмні продукти та системи.

Одержані результати дають підстави вважати, що мета диплому реалізована, основні цілі досягнуті.

Література

1. Wi-Fi следит за тобой, или Wi-Fi как система мониторинга [Електронний ресурс]. Хабр, 2016. – URL: https://habr.com/ru/post/399149/.
2. Location Technology and Intelligence [Електронний ресурс]. Skyhook, 2020. – URL: https://www.skyhook.com/.
3. Wi-Fi positioning system [Електронний ресурс]. Wikipedia, 2020. – URL: https://en.wikipedia.org/wiki/Wi-Fi_positioning_system.
4. Воронцов Ю.А., Козинец А.В. Стандарты веб-сервисов для создания распределенных информационных систем / Ю.А.Воронцов, А.В.Козинец, // Век качества. 2015. №3.
5. JavaSpript. Подробное руководство / Под ред. Дэвид Флэнаган. СПб, СимволПлюс, 2008. – 923 с.
6. "Svelte@3.20.1". [Електронний ресурс]. – Режим доступу: https://bundlephobia.com/result?p=svelte@3.20.1. BundlePhobia. March 22, 2020
7. Изучаем Node.js. - М.: Питер, 2013. - 400 с.
8. Кантелон, М. Node.js в действии / М. Кантелон. - М.: Питер, 2015. - 810 с.
9. Сухов, Кирилл Node.js. Путеводитель по технологии / Кирилл Сухов. - М.: ДМК Пресс, 2015. - 416 с.
10. Хэррон, Дэвид Node.js Разработка серверных веб-приложений на JavaScript / Дэвид Хэррон. - М.: ДМК Пресс, 2014. - 144 с.
11. Хэррон, Дэвид Node.js Разработка серверных веб-приложений на JavaScript / Дэвид Хэррон. - Москва: СПб. [и др.] : Питер, 2014. - 144 с.
12. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
13. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.
14. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

15. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
16. Гэддис Т. Начинаем программировать на Python. – 4-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 768 с.
17. Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768 с.
18. Свейгарт, Эл. Автоматизация рутиных задач с помощью Python: практическое руководство для начинающих. Пер. с англ. — М.: Вильямс, 2016. – 592 с.
19. Рейтц К., Шлюссер Т. Автостопом по Python. – СПб.: Питер, 2017. – 336 с.: ил. – (Серия «Бестселлеры O'Reilly»).
20. Любанович Билл Простой Python. Современный стиль программирования. – СПб.: Питер, 2016. – 480 с.: – (Серия «Бестсепперы O'Reilly»).

ДОДАТОК 1

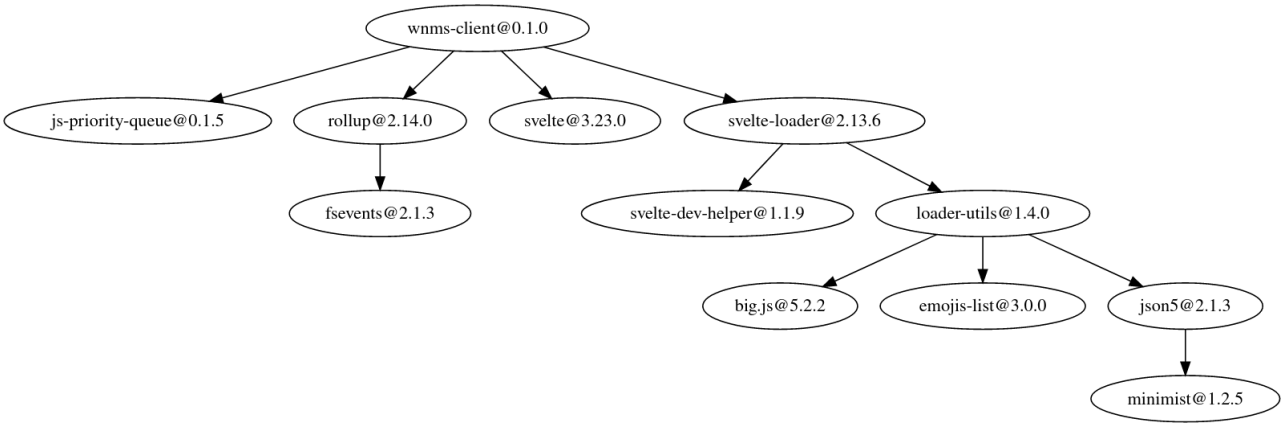
Система моніторингу користувачів бездротових мереж Wi-Fi

Граф залежності клієнтського додатку

ІАЛЦ.467200.004 Д1

Листів 1

2020



					ІАЛЦ.467200.004 Д1		
Змн.		ПІБ	Підпис	Дата			
Розробив		Булах А.О.			Система моніторингу користувачів бездротових мереж Wi-Fi. Додаток 1	Літ.	Арк. 1 / Акрушів 1
Перевірив		Алещенко О.В					
						КПІ ім. Ігоря Сікорського ФІОТ ІО-63	
Н/Контр.		Сімоненко В.П.					
Зав.каф.		Стіренко С.Г.					

ДОДАТОК 2

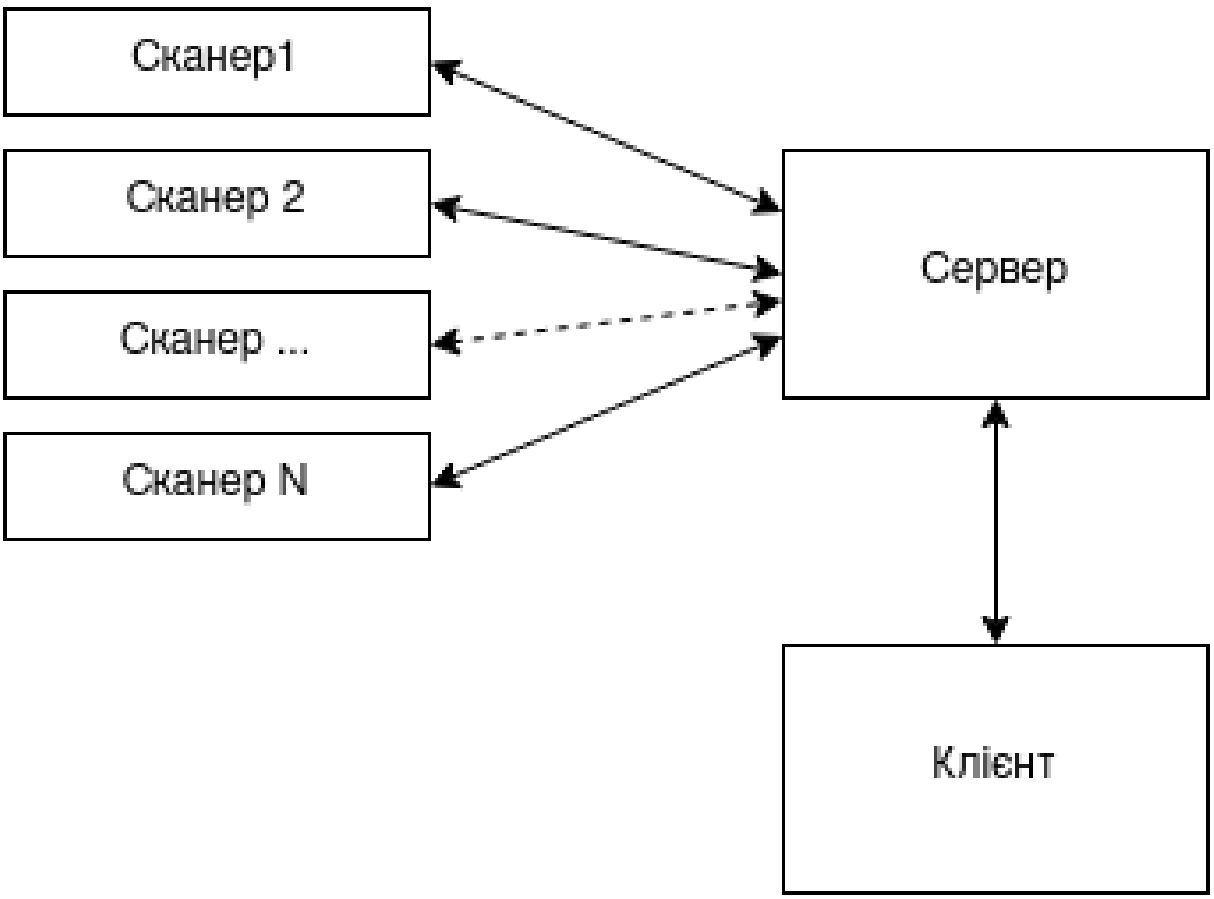
Система моніторингу користувачів бездротових мереж Wi-Fi

Структурна схема системи

ІАЛЦ.467200.005 Д2

Листів 1

2020



					ІАЛЦ.467200.005 Д2		
Змн.		ПІБ	Підпис	Дата			
Розробив		Булах А.О.			Система моніторингу користувачів бездротових мереж Wi-Fi. Додаток 2	Літ.	Арк. 1 / Акрушів 1
Перевірив		Алещенко О.В					
						КПІ ім. Ігоря Сікорського ФІОТ ІО-63	
Н/Контр.		Сімоненко В.П.					
Зав.каф.		Стіренко С.Г.					

ДОДАТОК З

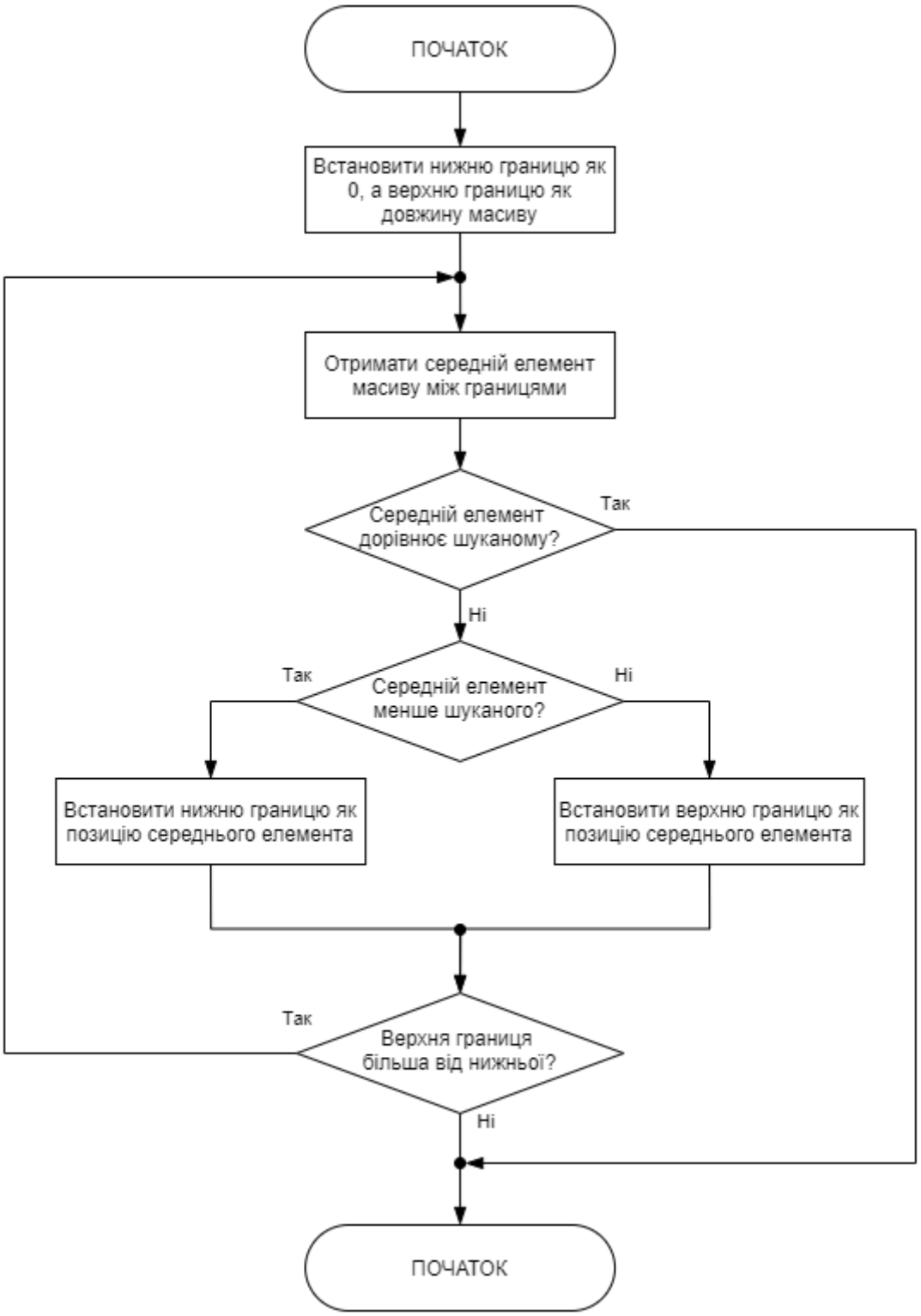
Система моніторингу користувачів бездротових мереж Wi-Fi

Алгоритм бінарного пошуку

ІАЛЦ.467200.006 Д3

Листів 1

2020



					ІАЛЦ.467200.006 Д3		
Змн.		*ПІБ*	*Підпис*	*Дата*			
Розробив		*Булах А.О.*			*Система моніторингу користувачів бездротових мереж Wi-Fi.* *Додаток 3*	*Літ.*	*Арк.* 1 / *Акрушів* 1
Перевірив		*Алещенко О.В*					
						КПІ ім. Ігоря Сікорського ФІОТ ІО-63	
Н/Контр.		*Сімоненко В.П.*					
Зав.каф.		*Стіренко С.Г.*					

ДОДАТОК 4

Система моніторингу користувачів бездротових мереж Wi-Fi

Лістинг програми скануючого пристрою

ІАЛЦ.467200.007 Д4

Листів 6

2020

```
#!/usr/bin/python3

import os
import socket
import sqlite3
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
from os import system
from sys import argv
from threading import Thread, Event
from time import sleep, time_ns

from scapy.layers.dot11 import Dot11FCS
from scapy.sendrecv import sniff

DB = 'data.db'

CREATE_TABLES_SQL = ('CREATE TABLE IF NOT EXISTS scans (ts
TIMESTAMP, mac CHAR(12)); CREATE TABLE IF NOT EXISTS sessions (start
TIMESTAMP, end TIMESTAMP);')

queue = []
streaming = False

server_running = Event()


def run_monitor(iface):
    print('configuring the interface...')

    system(f'ip link set {iface} down')
    system(f'iw dev {iface} set type monitor')
    system(f'ip link set {iface} up')

    print('running the monitor...')

    conn = sqlite3.connect(DB)
    c = conn.cursor()
    c.executescript(CREATE_TABLES_SQL)
    conn.commit()

    session_start = int(time_ns() / 1e6)

    def handler(p):
        if p.haslayer(Dot11FCS) and p.type == 0 and p.subtype in (0, 2, 4):
            ts = int(time_ns() // 1e6)
```

```
            mac = p.addr2.replace(':', '')

            c.execute('INSERT INTO scans VALUES (?, ?)', (ts, mac))
            conn.commit()

            if streaming:
                queue.append(f'{{"ts":{ts},"mac":"{mac}"}}')

    try:
        sniff(iface=iface, prn=handler)
    except KeyboardInterrupt:
        system(f'ip link set {iface} down')
        raise
    finally:
        c.execute('INSERT INTO sessions VALUES (?, ?)', (session_start, int(time_ns()
/ 1e6)))
        conn.commit()
        conn.close()


def run_event_server():
    class HttpHandler(BaseHTTPRequestHandler):

        def version_string(self):
            return 'WNMS/1.0.0'

        def handle_one_request(self):
            try:
                super().handle_one_request()
            except BrokenPipeError:
                self.log_error('broken pipe')
            global streaming
            streaming = False

        def do_POST(self):
            if self.path != '/reset':
                self.send_error(404)
                return

            self.send_response(200)
            self.send_header('Connection', 'keep-alive')
            self.send_header('Content-Length', '0')
            self.end_headers()

            conn = sqlite3.connect(DB)
```

```
            c = conn.cursor()
            c.executescript('DROP TABLE scans;'
                            'DROP TABLE sessions;'
                            '{}'.format(CREATE_TABLES_SQL))
            conn.commit()
            conn.close()

        def do_GET(self):
            if self.path == '/sessions':
                self.send_response(200)
                self.send_header('Content-Type', 'application/json')
                self.send_header('Connection', 'keep-alive')

                conn = sqlite3.connect(DB)
                c = conn.cursor()
                c.execute('SELECT * FROM sessions')
                json = '[' + ','.join(f'{{"start":{start},"end":{end}}}' for (start, end) in
c.fetchall()) + ']'
                conn.commit()
                conn.close()

                self.send_header('Content-Length', str(len(json)))
                self.end_headers()

                self.wfile.write(bytes(json, 'ascii'))
                return

            if self.path.startswith('/get'):
                path_part = self.path[4:]
                if len(path_part) == 0:
                    start, end = -1, -1
                else:
                    if not path_part.startswith('/'):
                        self.send_error(400, explain='malformed range')
                        return
                    params = path_part[1:].split('-', 1)
                    if len(params) != 2:
                        self.send_error(400, explain='malformed range')
                        return
                    try:
                        start, end = map(int, params)
                    except ValueError:
                        self.send_error(400, explain='start or end of the range are not
integers')
                        return
```

```
                if end < 0:
                    self.send_error(400, explain='malformed range')
                    return
                if end < start:
                    self.send_error(400, explain='range end is less than range start')
                    return

            self.send_response(200)
            self.send_header('Content-Type', 'application/json')
            self.send_header('Connection', 'keep-alive')

            conn = sqlite3.connect(DB)
            c = conn.cursor()
            if start == -1:
                c.execute('SELECT * FROM scans')
            else:
                c.execute('SELECT * FROM scans WHERE ts BETWEEN ? and ?',
(start, end))
            json = '[' + ','.join(f'{{"ts":{ts},"mac":"{mac}"}}' for (ts, mac) in
c.fetchall()) + ']'
            conn.close()

            self.send_header('Content-Length', str(len(json)))
            self.end_headers()

            self.wfile.write(bytes(json, 'ascii'))
            return

        if self.path == '/listen':
            self.send_response(200)
            self.send_header('Content-Type', 'text/event-stream')
            self.send_header('Cache-Control', 'no-cache')
            self.send_header('Connection', 'keep-alive')
            self.end_headers()

            global queue, streaming
            streaming = True
            queue = []

            last_msg = time_ns()
            event_id = 0

            # yeah just short-polling the queue each 50ms
            # also sending a ping event each 20 seconds that had no events
            while not self.wfile.closed:
```

```
                now = time_ns()

                if len(queue) != 0:
                    moved, queue = queue, []
                    for data in moved:
                        self.wfile.write(bytes('event: scan\nid:{}\ndata:
{}\n\n'.format(event_id, data), 'utf-8'))
                        event_id += 1
                    self.wfile.flush()
                    last_msg = now

                if now - last_msg > 20e9:
                    last_msg = now
                    self.wfile.write(b'event: ping\n\n')
                    self.wfile.flush()

                sleep(0.05)
            return

        self.send_error(404)

    class Server(ThreadingHTTPServer):

        def server_activate(self):
            super().server_activate()
            server_running.set()

    print('running HTTP server...')
    Server(('', 8000), HttpHandler).serve_forever()
    print('ran the server lol')


def run_discovery_broadcast():
    server_running.wait()

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)

    msg = bytes('[WNMS]' + socket.gethostname(), 'utf-8')

    while True:
        sock.sendto(msg, ('255.255.255.255', 1717))
        sleep(0.5)
```

```
def main():
    arg = ''
    try:
        arg = argv[1]
    except IndexError:
        print(f'usage: {argv[0]} <iface>')
        exit(1)

    if os.getuid() != 0:
        print('this program requires root permissions for setting up the monitor')
        exit(1)

    Thread(target=run_event_server, daemon=True).start()
    Thread(target=run_discovery_broadcast, daemon=True).start()
    run_monitor(arg)


if __name__ == '__main__':
    main()
```